2-1-1997

A Morphological array image processor controller chip set

Christopher Insalaco

A MORPHOLOGICAL ARRAY IMAGE PROCESSOR CONTROLLER CHIP SET

by

Christopher J. Insalaco

A thesis submitted
in
partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Engineering

Approved by: Prof. _____
George A. Brown, Thesis Advisor

Prof._____
Robert E. Pearson

Prof._____
Roy S. Czernikowski, Department Head

Department of Computer Engineering
College of Engineering
Rochester Institute of Technology
Rochester, New York
February, 1997

A MORPHOLOGICAL ARRAY
IMAGE PROCESSOR
CONTROLLER CHIP SET

I Christopher J. Insalaco hereby **grant permission** to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

2/13/97 _____

Date Christopher J. Insalaco

Rochester Institute of Technology

Abstract

A MORPHOLOGICAL ARRAY
IMAGE PROCESSOR
CONTROLLER CHIP SET

by Christopher J. Insalaco

The design, simulation and layout of a controller chip set for a morphological array image processor shall be discussed. These VLSI chips in conjunction with the Morphological Array Processor (MAP) and Arithmetic Logic Unit (ALU) chip sets perform the morphological image processing operations of erosion and dilation on 512x512 pixel, 8-bit gray scale images using a 7x7 windowing matrix in real time (60 frames per second). The controller chip set design allows for pipelining of successive MAP's as well as operation on 1024x1024 pixel, 8-bit gray scale images.

To facilitate the design, additional scaleable CMOS standard library cells and corresponding parameterized schematic library components were designed and integrated with the RIT CMOS standard cell library designed by Computer Engineering graduate student Larry Rubin as part of his Masters thesis[1]. In particular, additional D flip-flops with both Q and Q bar outputs, and-or-inverts, or-and-inverts, CMUXes, and MOSIS 64 and 84 pin pad rings were created. The cells were designed to be fabricated using the Metal Oxide Semiconductor Implementation System (MOSIS) scaleable CMOS 2.0 μm N-well (SCN) process. A complete set of Cadence design rule verification tools were also integrated with the existing CAE tool set to perform design rule checking (DRC), electrical rule checking (ERC), layout versus schematic checking (LVS), and layout parameter extraction (LPE) for the MOSIS SCN 2.0 μm N-well dense rule set. To verify the CMOS standard cell designs, test

chips were designed and sent to MOSIS for fabrication. The layout and design rule verification of the final two test chips, test chips five and six, was performed by the author. Test chip four contains a variety of MUXes and D flip-flops, test chip five contains a variety of transfer gates and inverters.

The controller chip set consists of a 64 pin control chip (Controller) and an 84 pin memory controller chip (Mem_Control). The controller chips provide the ability to selectively process 512x512 or 1024x1024 image sizes by modifying the pullup or pulldown of a "size" bit. A selectable delay was implemented, through the pullup or pulldown setup of three delay bits, in the Controller to allow the Controller to be used with the single chip VLSI MAP design, the seven chip VLSI MAP design, and the Actel gate array MAP. The controller chip set allows successive MAPs to be pipelined by connecting the next Controllers pipeline start pin to the previous stages pipeline start next pin.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

To Linda and Michelle, who make it all worthwhile.

The author wishes to thank Jeff Correll for his help with the cell library and the incorporation of the REMEDI design rule checker into the Chipgraph macros, Shishir Ghate for his help with the cell library and ALU design and the integration of the library with Cellstation, Jeff Hanzlik for his help with the logical definition of the MAP, Jens Rodenberg for the MAP Architecture section and the logical design of the Controller, and Larry Rubin for his work in designing the cell library, enhancing the design automation tools and the design of the MAP.

LIST OF ABBREVIATIONS

ALU	Arithmetic Logic Unit
CAD	Computer Aided Design
CIF	Caltech Intermediate Form
CPF	Circuit Path Finder
DRC	Design Rule Checker
ERC	Electrical Rule Checker
HDL	Hardware Description Language
IBM	International Business Machines Inc.
LPE	Layout Parameter Extractor
LVS	Layout Versus Schematic
LSI	Large Scale Integration
MAP	Morphological Array Processor
MOSIS	Metal Oxide Semiconductor Implementation System
MSI	Medium Scale Integration
NSA	National Security Agency
RIT	Rochester Institute of Technology
SSI	Small Scale Integration
VLSI	Very Large Scale Integration

Accusim, Cellgraph, Chipgraph, Expand, Extract, IDEA Station, MCIF, Mspice, Neted, REMEDI, and Symed are registered trademarks of Mentor Graphics Corporation.

Dracula II is a registered trademark of Cadence Inc.

TekWAVES and LV500 are registered trademarks of Tektronics Inc.

I. Introduction

To verify the design of the RIT CMOS standard cell library, created as a part of Larry Rubin's thesis [1], it became necessary to set up layout verification tools for the MOSIS 2μm N-well CMOS process. It was decided that the MOSIS dense rule set would be used to allow for more compacted design layouts. So, a set of rules checkers to automate the design verification for this process became necessary. The Cadence Dracula II tool set will be used in this work; specifically the Cadence design rule checker (DRC), electrical rule checker (ERC), layout versus schematic (LVS), and layout parameter extractor (LPE) products. A technology (".com") file must be defined for the MOSIS dense rule set for each of the products in the Cadence tool set.

Additional RIT CMOS standard cells will be designed, layed out, and verified as necessary to complete the morphological array controller chip set. In addition test chips will be layed out and sent to MOSIS for fabrication to test some of the standard cell designs.

Once the verification tools and additional RIT CMOS standard cells were defined, they were utilized to create a morphological array controller chip set for use with the morphological array processor (MAP) chip set designed by Larry Rubin, and the pre and post ALU chip set designed by Shishir Ghate[2]. The following are the design goals for the controller chip set:

1. to match the functionality of the ACTEL controller circuitry design of Jens Rodenberg[3]

2. to allow real time image processing of up to 60 frames per second at a 16 MHz clock speed

3. to work in conjunction with the timing and control signals of the MAP and ALU chip sets to allow the morphological image processing operations of erosion and dilation on 512x512 or 1024x1024 pixel nine-bit gray scale images with a 7x7 window

4. to allow future designs using the controller and MAP to be pipelined

5. to allow future designs to have more than four memory units

6. to fit the controller chip set onto MOSIS standard 64 and 84 pin pad ring chips

II. Historical Review

A. Design Before the Library

Without the RIT CMOS standard cell library a designer would have had to enter two different sets of schematics to do both logical and circuit level simulations. This is both inefficient and unreliable, as there is no mechanism to verify the logic level schematic with the equivalent transistor level schematic.

Another drawback of the generic logic level library (gen_lib) supplied by Mentor Graphics, is that every component has a default delay of zero when simulated. This is overly optimistic, and prevents operation in systems with feedback, such as an SR latch. Race conditions are not modeled, nor are settling glitches of the circuit's outputs. The library solves these problems by simply having all components default to at least one unit delay rise and fall time when logically simulated.

Before the library was created, no standardized mechanism existed to actually fabricate designs through the MOSIS program. The SPICE files were obsolete, and overly simplified (they did not model leakage current or junction capacitance) and there were no pad cells.

For the design and implementation of any VLSI project, especially if it is intended to be fabricated, the compilation of a standard library is not only practical, but a necessary and an inevitable evolutionary stage of any CAD center.

B. Test Chips

At the time the RIT cell library was proposed, MOSIS supported twelve different processes, including 2.0, 1.6, and 1.2 micron CMOS and even a GaAs process. A standard cell library (with no components) available from MOSIS, was developed at the University of Mississippi for the National Security Agency. This cell library is publicly available. When the RIT cell library was proposed, Larry Rubin designed and fabricated a test chip to compare some prototype cells with the equivalent cells supplied by the NSA to determine if there were any advantages to generating a complete library from scratch[1]. The alternative would have been to create a matching component library for the NSA cell library.

The test chip consisted of MUXes, D flip-flops, and an 8-bit adder. The MUXes compared the RIT nMOS and CMOS transmission gate MUXes against the NSA MUX called "dsel". A D flip-flop with asynchronous resets from each library was used. For overall speed comparison, two 41-stage ring oscillators were created.

The RIT cells required 1/3 to 1/2 the area of the equivalent NSA cells, and had less input capacitance, while having the same drive capabilities. Because of this, the RIT cells were faster. Due to less than maximal contacting and unnecessarily thin power routing in the NSA cells (causing unnecessary internal resistance), the RIT cells were calculated to consume less power.

Another advantage of the RIT cells is that there are several drive sizes available for each component. The inverter, for example, has nine different drive sizes for the designer to choose from. The NSA library offers two sizes of inverter, and only one drive size was available for most of the cells. After all of these considerations were taken into account, it was obvious that a more useful library would result by developing it from scratch.

The test chip has the distinction of being the first RIT design to be processed at MOSIS.

To test the functionality of the cell library, 5 additional test chips were designed and fabricated by Shishir Ghate[2]. Each component on the test chips has its own output pad. This was done to allow for more accurate results from the testing. One could have conceivably placed many more components inside each test chip and multiplexed the outputs. Although this would have allowed more components to be tested per chip, it was determined that the multiplexing would have been detrimental to the final cause of obtaining accurate results.

The number of components that each input pad is driving was kept to a value of ten or less. A value of ten was chosen because the capacitive load produced by these gates would be small. This was to insure that results obtained would not be diminished by overloading the input pads.

C. The Commission of a Morphological Chip Set

In April 1990, Professor Edward Dougherty from the Center for Imaging Science discussed morphology and the morphologic operations of erosions and dilations with the faculty in the Computer Engineering Department.

In June 1990, Computer Engineering graduate student Jeff Hanzlik was selected as principal investigator for the exploration of a prototype morphologic platform. It was decided to implement a 7x7 morphological array processor on a printed circuit board that could be inserted into an IBM PC/AT compatible machine and operate on 512x512 pixel 8-bit gray scale images[4].

In September of 1990, Computer Engineering graduate student Jens Rodenberg joined the project. Initially working with simulations of morphological operations he developed a new architecture by November, whereby the undefined pixels (represented mathematically by negative infinity) were implicitly defined via control bits, as opposed to the original architecture that required explicit negative infinities in the data path. The new architecture also employed enhanced pipelining, thereby increasing the efficiency of the processor[3].

An initial goal was to fit the prototype onto an AT standard card using ACTEL gate array chips. This proved to be impossible, as the Actel gate array chips that were used did not allow a high enough level of integration. It was also originally desired to have the prototype operate in real time. Again, due to Actel speed limitations, the technology prevented the desired performance from being achieved. In order to meet these goals, full custom VLSI was required.

In 1991 Computer Engineering graduate student Lawrence Rubin joined the project with the goal of designing a single VLSI chip Morphological Array Processor that would replace 23 of the Actel gate arrays for the processor array itself[1]. Later, Computer Engineering graduate student Chris Insalaco joined the project to design a smaller more integrated 2-chip set Controller to replace 5 Actel gate arrays of the control logic, while also increasing performance. Also Computer Engineering graduate student Shishir Gate joined the project to design the pre and post Arithmetic Logic Units for the MAP chip set to complete the VLSI design[2].

III. Theory

A. Morphology

Morphological image processing refers to the analysis and processing of an image based upon a knowledge of its structure or form with the intent of modifying that form or structure. The theoretical basis for morphological image processing dates back to the work of H. Minkowski on spatial set algebra. Based upon this work, G. Matheron and J. Serra of France and S. Sternberg of the U.S. have developed a set-theoretic approach to image processing[5,6]. In this approach, binary images are treated as sets in a background space that can then be acted upon by the usual set operations of union and intersection, in conjunction with another image set called a structuring element. The structuring element is used to probe or fit the image to extract information about its shape.

For gray scale images the set operations of union and intersection become the maximum and the minimum functions, respectively.

The two fundamental morphological operations are dilation and erosion. Dilation of an image yields an image that is uniformly larger than the original image while erosion of an image yields a uniformly smaller image. Building upon the basic morphological operations, a large variety of morphological filters can be generated for such applications as edge detection, segmentation, and enhancement of images. Morphological filters can also be used instead of standard linear filters. The basic morphological operators will be derived using the Euclidean model, then the digital and gray scale models will be presented.

1. EUCLIDEAN MORPHOLOGY

Morphological operations are built upon the set operations union and intersection as well as the translation and reflection operations. Given an image A in R^2 the translation of A by the point x in R^2 is defined by:

$T_xA = \{a + x : a \in A\}$ where the plus sign refers to vector addition. (1)

Considering the point x to be a vector in the plane, T_xA is A translated along the vector x. A reflected image, $B^\wedge$ in R^2 is one that has been rotated 180° around the origin or alternatively one that has been flipped from left to right and from top to bottom. The first fundamental operation in morphological analysis is Minkowski addition. Given two images A and B in R^2 , the Minkowski sum is defined as:

$$A \oplus B = \bigcup_{b \in B} T_bA = \bigcup_{b \in B} \{a + b : a \in A\} \quad (2)$$

$A \oplus B$ is constructed by translating A by each element of B and then taking the union of all the resulting translates. The second fundamental morphological operation is Minkowski subtraction. Given two images A and B in R^2, Minkowski subtraction is defined as:

$$A \ominus B = \bigcap_{b \in B} T_bA = \bigcap_{b \in B} \{a + b : a \in A\} \quad (3)$$

$A \ominus B$ is constructed by translating A by every element of B then the intersection of the resulting translates is taken.

In morphological processing, the Minkowski sum is referred to as a dilation, and denoted as:

$$D(A,B) = A \oplus B \tag{4}$$

The morphological operation erosion has two different definitions in the literature. Serra defines erosion as Minkowski subtraction[6], denoted as:

$$E(A,B) = A \ominus B \tag{5}$$

whereas Sternberg and Matheron [6] define erosion as:

$$E(A,B) = A \ominus B\hat{} = \bigcap_{b \in B\hat{}} T_b A \tag{6}$$

This second definition of erosion will be used for our purposes as it lends itself better to a digital implementation. The second image B is generally referred to as the structuring element. If $B = B\hat{}$, as it usually does, then erosion again becomes equal to Minkowski subtraction. Another form of the Minkowski subtraction equation is very useful in image processing, it states that the Minkowski subtraction of B from A is composed of all elements x in R^2 such that B translated by x is a subset of A, where $B \neq \varnothing$. This is denoted as:

$$A \ominus B = \{x : T_x B\hat{} \subset A\} \tag{7}$$

This equation can also be written as:

$$A \ominus B\hat{} = \{x : T_x B \subset A\} \tag{8}$$

An alternative definition of erosion can then be written as:

$$E(A,B) = \{x : T_x B \subset A\} \tag{9}$$

Another form of the Minkowski addition (dilation) is also very useful. It states that the Minkowski sum is composed of all elements x in R^2 such that B^ translated by x is not a subset of A, where $B \neq \varnothing$. This is denoted as:

$$D(A,B) = \{x : T_x B^{\wedge} \not\subset A\} \tag{10}$$

These are the forms of the erosion and dilation equations that are used in the digital implementation.

Erosion and dilation are often performed in succession on an image. A dilation followed by an erosion is called a "closing" operation and is denoted:

$$C(A,B) = E(D(A,B^{\wedge}), B^{\wedge}) = (A \oplus B^{\wedge}) \ominus B \tag{11}$$

An erosion followed by a dilation is called an "open" operation and is denoted by:

$$O(A,B) = D(E(A,B), B) = (A \ominus B^{\wedge}) \oplus B \tag{12}$$

Some of the properties of Minkowski addition, Minkowski subtraction, opening and closing will now be presented.

$$A \oplus B = B \oplus A \qquad \text{addition is commutative} \tag{13}$$

$$A \ominus B \neq B \ominus A \qquad \text{subtraction is not commutative} \tag{14}$$

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) \qquad \text{addition is associative} \tag{15}$$

Associative addition allows structuring elements to be decomposed and chained to achieve the effect of larger structuring elements from small structuring elements.

$$(A \oplus B)^c = A^c \ominus B \tag{16}$$

$$(A \ominus B)^c = A^c \oplus B \tag{17}$$

Dilation and erosion are dual properties, the dilation of the background of an object is the same as the erosion of the object, and vice versa.

$$\text{if } A \subseteq B \text{ then } A \oplus B \subseteq B \oplus C \tag{18}$$

$$\text{if } A \subseteq B \text{ then } A \ominus B \subseteq B \ominus C \tag{19}$$

Addition and subtraction are both increasing operations.

$$A \oplus (T_x B) = T_x(A \oplus B) \tag{20}$$

$$A \otimes (T_x B) = (T_x A) \ominus B = T_x(A \ominus B) \tag{21}$$

Addition and subtraction are both translation invariant.

$$O(A,B) \subseteq A \tag{22}$$

Open is antiextensive, the open operation tends to decrease the spatial extent of an image.

$$C(A,B) \supseteq A \tag{23}$$

Close is extensive, the close operation tends to increase the spatial extent of an image.

$$O(O(A,B), B) = O(A,B) \tag{24}$$

$$C(C(A,B), B) = C(A,B) \tag{25}$$

Open and close are idempotent (repeated applications have no effect).

$$\text{if } A \subset F \text{ then } O(A,B) \subset O(F,B) \tag{26}$$

$$\text{if } A \subset F \text{ then } C(A,B) \subset C(F,B) \tag{27}$$

Open and close are increasing operations.

2. DIGITAL MORPHOLOGY

Let F (j,k) be a binary valued image matrix. A pixel at coordinate (j,k) is an element of image F (j,k) if and only if it is a logical one. An image B (j,k) is a subset of an image A (j,k) if for every logical one pixel in B (j,k) there is a logical one pixel in A (j,k). An image F^c (j,k) is the complement of F (j,k) if all the pixels in F^c (j,k) are the opposite logically of those in F (j,k). A reflected image F^ (j,k) is formed by flipping the image matrix F (j,k) from left to right and from top to bottom. Translation of an image consists of shifting the image by r rows and c columns from the origin. This is denoted as:

$$G(j,k) = T_{r,c}\{F(j,k)\} \tag{28}$$

For the following definitions, assume an NxN image matrix F (j,k) and an LxL, where L is odd, structuring element matrix H (j,k). Minkowski addition is defined, similar to the Euclidean case, as:

$$G(j,k) = \bigcup_{(r,c)\in H} T_{r,c}\{F(j,k)\} \tag{29}$$

The resulting image size is MxM where M = N + L 1. The definition of dilation is then:

$$G(j,k) = F(j,k) \oplus H(j,k) \tag{30}$$

Minkowski subtraction is similarly defined as:

$$G(j,k) = \bigcap_{(r,c)\in H} T_{r,c}\{F(j,k)\} \tag{31}$$

and the definition of erosion is then:

$$G(j,k) = F(j,k) \ominus H\hat{}(j,k) = \bigcap_{(r,c)\in H\hat{}} T_{r,c}\{F(j,k)\} \tag{32}$$

Another definition of dilation, analogous to the alternate definition in the Euclidean model, based on the scanning and processing of H (j,k) over F (j,k) is:

$$G(j,k) = \bigcup_{(m,n)} \{F(m,n) \cap H(j-m+S, k-n+S)\} \tag{33}$$

where $\max[1, j-Q] \le m \le \min[N, j+Q]$ and $\max[1, k-Q] \le n \le \min[N, k+Q]$ and $S = (L + 1)/2$ and $Q = (L - 1)/2$.

The alternate definition for erosion is similarly defined as:

$$G(j,k) = \bigcap_{(m,n)} \{F(m,n) \cup H(j-m+S, k-n+S)\} \tag{34}$$

where the same limits apply as for dilation. The digital opening and closing equations can also be defined analogously to the corresponding Euclidean operators as:

$$O(F,H) = D(E(F,H), H) = [F(j,k) \ominus H\hat{}(j,k)] \oplus H(j,k) \tag{35}$$

$$C(F,H) = E(D(F,H^{\wedge}), H^{\wedge}) = [F(j,k) \oplus H^{\wedge}(j,k)] \ominus H(j,k) \qquad (36)$$

3. GRAY SCALE MORPHOLOGY

Let F (j,k) be a gray scale image quantized to an arbitrary number of gray levels, n, whose pixel values can be between zero and 2^n - 1 or undefined, (*). Only defined pixels may be elements of the gray scale image. An undefined pixel may come about by a variety of methods. It may be due to a sensor problem or the pixel may simply be out of the bounds of the image. For gray scale images the union operation is interpreted as the maximum of the two input images taken point by point. The intersection operation is interpreted as the minimum of the two images taken point by point. The usual maximum and minimum definitions can be extended to include undefined pixel values by treating (*) as if it were negative infinity. The max of two defined pixels is the highest value pixel, the max of a defined and an undefined pixel is the defined pixel value, the max of two undefined pixels is (*). The min of two defined pixels is the lowest pixel value, the min of a defined and an undefined pixel is (*), and the min of two undefined pixels is (*). The complement F^c (j,k) of a gray scale image F (j,k) is found by replacing every pixel value, i, with 2^n 1 - i. The translation, and reflection of a gray scale image are defined and denoted the same as they are for a binary image. For an NxN image matrix F (j,k) and an LxL, where L is odd, structuring element matrix H (j,k), gray scale Minkowski addition is defined as:

$$G(j,k) = \max_{(r,c)\in H} [T_{r,c}\{F(j,k)\}] \qquad (37)$$

and gray scale Minkowski subtraction as:

$$G(j,k) = \min_{(r,c) \in H} [T_{r,c} \{F(j,k)\}] \quad (38)$$

The alternate form of the dilation operation is defined as:

$$G(j,k) = \max [F(m,n) + H(j-m+S, k-n+S)] \quad (39)$$

The alternate form of the erosion operation is defined as:

$$G(j,k) = \min [F(m,n) + H(j-m+S, k-n+S)] \quad (40)$$

where the same limits as above for binary images hold. It may be of interest to compare the above equation with the defining equation for convolution. In the dilation and erosion equations the max or min operations are analogous to the summations in the convolution equation and the point by point additions are analogous to the point by point multiplication's in the convolution equation. Similar to convolution, dilation can be thought of as the scanning and processing of F (j,k) by H (j,k) rotated by 180° . The gray scale opening and closing operations are once again defined as:

$$O(F,H) = D(E(F,H), H) = [F(j,k) \ominus H\hat{}(j,k)] \oplus H(j,k) \quad (41)$$

$$C(F,H) = E(D(F,H\hat{}), H\hat{}) = [F(j,k) \oplus H\hat{}(j,k)] \ominus H(j,k) \quad (42)$$

where now gray scale erosions and dilations are performed. Because the max and min operations involved in erosion and dilation can be performed sequentially, the erosion and dilation operations can be decomposed into a series of iterative erosions or dilations to accomplish the same effect as a larger structuring element. Three iterations of a morphological image processing operation using a 3x3 structuring element accomplishes the same effect as one operation using a 7x7 structuring element.

B. Design Rule Checking

A design rule checker checks that the widths, spacings, and overlaps of the features in a layout meet some minimum rules, the design rules. These design rules are due to the limitations in the integrated circuit manufacturing process. Many different mechanisms can contribute to deviations in feature shapes such as electromigration, mask misalignment, overetching, variations in photoresist exposure, and the spread of diffusion and implant regions around transistors. The purpose of design rules is to attempt to guarantee that, under the accumulated set of process variations, the circuit continues to function as designed. The Mead and Conway design rules are specified in relative terms using a scaleable unit, lambda (λ), instead of distances such as microns[7]. These design rules are conservative enough to allow fabrication at a smaller geometry by simply changing the value of λ. Using the most conservative value for each design rule, a simple set of spacing rules is produced. If layout density is important, then a larger variety of design rules must be considered. There is a trade-off of increased design checking time for denser rule sets because more complicated design rules involve more checks.

Design rules are constraints imposed on the geometry of layouts expressed as minimum separations and minimum sizes of layout features. Design checks are specified between both drawn layers and derived layers, that is layers that are made up of combinations of the drawn layers. For example, transistors are created at the areas where polysilicon and diffusion overlap. The analysis software can perform the Boolean operations AND, OR, and NOT on these layers to create derived layers, such as a transistor layer. The more derived

layers that are defined, the greater the design checking time and memory utilization.

Integrated circuit layouts consist of a collection of polygons of various colors representing features in different mask levels. Design rules generally specify rules on the edges of the polygons, instead of on the polygons themselves. One method of rule checking is to break polygons and rectangles down into their edges and perform rule checking on the edges. This initially increases the amount of data objects as there are more edges than polygons. But, it turns out that half of the edges can be removed from the edges data without loss of information by adding a direction bit to indicate which side of the edge is the inside area. Because almost all layouts are orthogonal, usually half of the edges can be eliminated.

Other methods of design checking include corner-based (or tile-based) checkers such as Magic and pixelmap (or raster) checkers[9]. Pixelmap checkers only work on Manhattan geometry. A small window is passed over the design and compared with a lookup table. The window size is a function of the grid spacing. For a layout with a small grid spacing, a larger window is required. Because the lookup table size grows, based on the window size, pixelmap checkers have large data requirements that make them impractical for big designs. Corner-based checkers are also limited to Manhattan geometry and are most often used as interactive solutions. These algorithms are based on the premise that a rule need only be checked at the corners, therefore most design rules can be verified by only checking the corners of the layout. Unfortunately, specifying the rules for a corner-based design checker is very difficult. We will discuss only edge based checkers for the rest of the thesis, as that is the method most frequently used by tools such as the Cadence Dracula II tool suite.

Layout analysis tools work on a "flat" representation of the layout, with all of the designs hierarchy levels removed. The first step of the design rule checker program is to read in the layout file and flatten the hierarchy by expanding the hierarchical instances. Next touching polygons on the same layer are merged, and then the polygons are converted to edges. The goal of merging the polygons is to remove unnecessary internal edges in the layout. To prevent false spacing rule violations from edges in the same node, when polygons are merged, a unique identification number is assigned to them. This number may be checked before applying spacing rules checks.

A common set of operations performed by design rules checkers on the flattened layout are bloating and shrinking. Bloating enlarges and shrinking contracts the layout by a uniform distance. To determine if two features on a layer are too close together, the layout can be bloated by half the design rule distance and re-examined. Any resulting overlap is a violation of the design rule limit. If the overlap polygon is expanded by slightly more than half the design rule, and ANDed with the original polygon edges, then the edges in error are highlighted (see Figure 1). Minimum feature sizes can be checked on a layer by shrinking by half the minimum size for the layer. Enclosure checks of one layer around another, such as contact cuts, can be calculated as a minimum spacing between the inside layer and the complement of the enclosing layer (see Figure 2). To check transistor gate overlap, another type of expansion is used. Each polysilicon edge is expanded in a perpendicular direction, then the overlap is checked (see Figure 3). To prevent erroneous minimum spacing violations from being detected, where the polysilicon and diffusion lines run into transistors, the expanded polysilicon area (calculated above for overlap) can be subtracted from the polysilicon lines before performing the minimum spacing check between the polysilicon and diffusion (see Figure 4)[11].

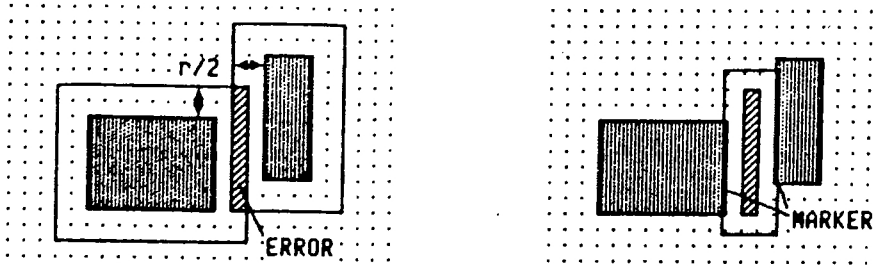

Figure 1 Spacing Check

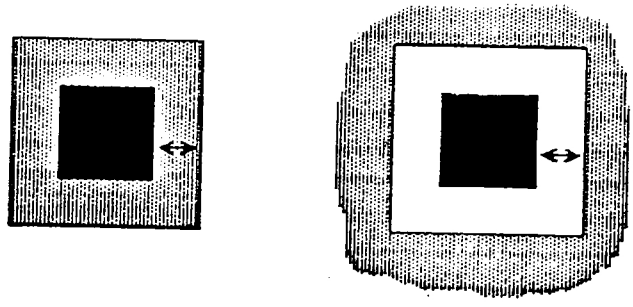

Figure 2 Enclosure Check

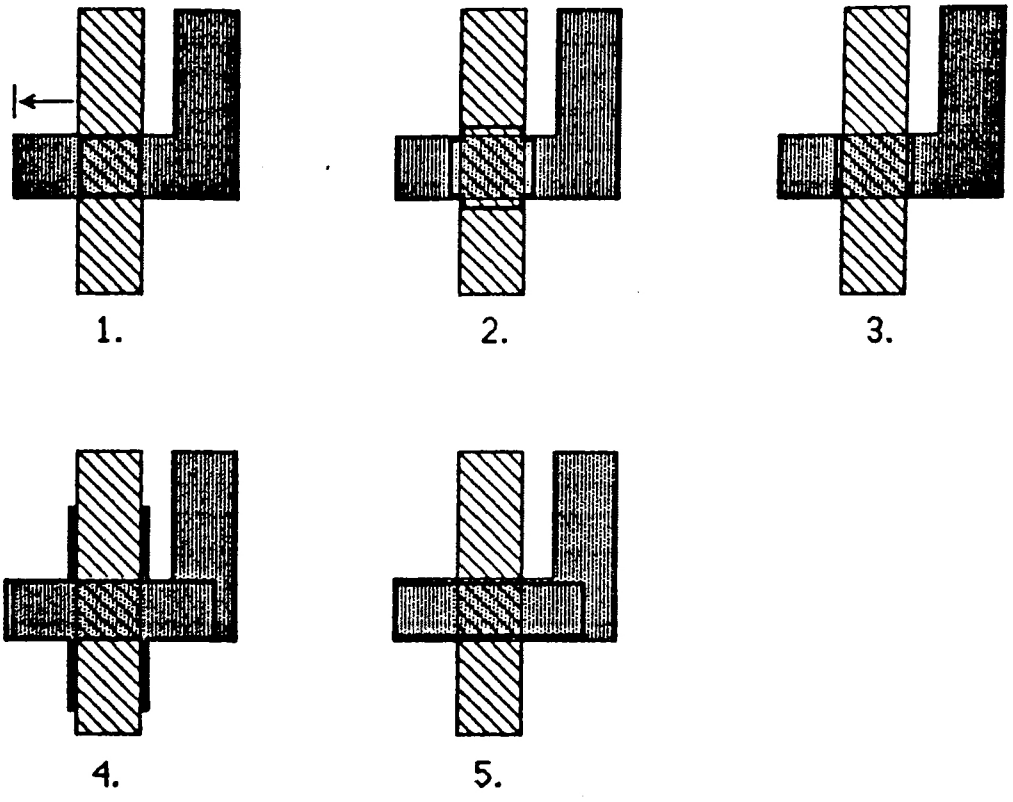

Figure 3 Transistor Overlap Check

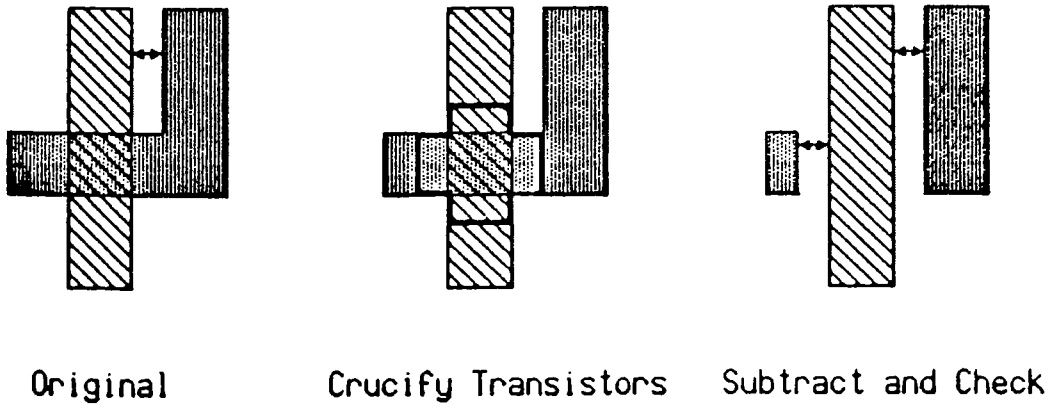

Figure 4 Transistor Spacing Check

Rules that check contact overlap and transistor spacing require Boolean operations to be performed on the input mask layers. It is simpler to first perform the Boolean operations on the layers to derive the transistor, transistor-overlap, and contact area pseudo-layers. Then the minimum width and spacing rules can be performed using these new pseudo-layers. The design rule check is therefore a two step operation, one step to derive the pseudo-layers, and one step to check the width and spacings of all the layers.

A design rule language is used to describe the derived pseudo-layers and the design rule specifications for the process. Three types of design checks can be defined: width checks, spacing checks, and enclosure checks. Width checks are defined by identifying the layer, the minimum width and the rule name; spacing checks are defined by identifying the two layers, the spacing distance, and the rule name; enclosure checks are defined by specifying the internal and external layers, the overlap, and the rule name. When the design rule checker is invoked, the definition file for the process is read in, and the derived pseudo-layers are created. On the second pass the design rules are checked.

The output error edges can be reported as a new error layer that can be overlaid on the original design layout. If a separate error cell is reported for each type of design rule error, then the user can view one type of error at a time, such as metal-to-metal spacing.

Circuit extraction consists of finding the transistors, the nets between the transistors, and the transistor areas and perimeters to calculate capacitance estimates and transistor size. The output is a list of transistors, their size and types, a netlist with the connections between their gates, sources and drains, and a list of capacitors from the nets to ground or between nets. The estimated capacitances can then be back annotated into the circuit design for more

accurate circuit simulations. The netlist generated can be compared with the circuit schematic to verify logic and schematic are equivalent.

In MOS circuits, transistors are created by crossing polysilicon and diffusion layers. Different types of transistors are detected by the presence or absence of implants and wells. The definitions of the pseudo-layers that comprise the different transistor types for a fabrication process are added to a technology file, similar to the design rule checker technology file. Once each transistor is uniquely identified and the nodes are numbered, the connections between the nodes are examined. Connections are preserved over continuous runs on a layer and through contacts connecting two layers. Incorrect transistor ratios and malformed transistors with floating nodes or shorts between power and ground can be identified with a simple electrical rules check. Transistor drive size is calculated by determining the width and length of the transistors. For rectangular transistors, the width is the one half the perimeter with the diffusion edge, and the length is one half the perimeter with the polysilicon edge.

Parameter extraction determines the electrical parameters from the layout for use in refining the circuit simulation timing. The two categories of capacitance that can be calculated during extraction are area and perimeter capacitance to substrate, and the capacitance between parallel wires or nodes. Each wiring layer and transistor type has a different capacitance factor, which is a function of the fabrication process. These factors are added to a technology definition file for the process, similar to the design rule checker technology file. During extraction, the areas and perimeter for each layer are calculated separately then multiplied by the appropriate scale factor, and the result is totaled to get a capacitance value for each node.

Wiring resistance can be calculated for long parallel lines as the length of the parallel lines, divided by the distance between them, times the resistance scale factor for the layer. Transistor resistance is proportional to the length over width ratio (the inverse of the Z-ratio) of the transistor, times the scale factor for the process. Generally, at the time of this project, most circuit extractors do not calculate wiring resistance for back annotating into the circuit design due to the large network of resistors that would result[9].

IV. Materials and Apparatus

A. Mentor and Cadence Tools

The additional standard cells for the RIT CMOS standard cell library and the morphological array controller chip set were designed on HP/Apollo workstations running the Aegis™ operating system. The RIT CMOS standard cell library has already been integrated with version 7.0 of the Mentor Graphics tool suite as a part of Larry Rubin's thesis[1]. The Mentor Graphics development tool suite consists of the Neted schematic capture tool, the Quicksim logic simulator, the Accusim SPICE circuit simulator, the Cellgraph cell automatic place and route tool, and the Chipgraph hand layout editing tool. In addition the Cadence Dracula II design rule checker (DRC), electrical rule checker (ERC), layout versus schematic (LVS), and layout parameter extraction (LPE) tool suite will be utilized. The Mentor Graphics tool suite is licensed and available on all of the HP/Apollo workstations in the lab. The Cadence tool suite is only licensed to run on two workstations within the lab.

When a top-down design methodology is used, the design process is divided into phases. The phases are design specification, functional design, logic design,

circuit design, and physical design. These design representations are used to describe different levels of abstraction of the system. The design automation process consists of a series of iterations of, and conversions between, these different representations of a system. Maintaining equivalency across the different design representations is an important issue for design automation tools.

The design specification stage is the manual process of specifying the system requirements. The behavior of the system, as a function of the inputs and outputs, is described during functional design. Logic and circuit design describe the logical or schematic structure of the system to match the functional design. Physical design involves converting the circuit structure into the format required to manufacture the physical device.

Each of the design phases may be composed of synthesis, analysis, and verification steps and therefore the design automation tools utilized during these steps can be categorized as synthesis, analysis, or verification and validation tools[9]. Synthesis tools generate a new representation of a design, such as the Mentor Graphics Cellgraph automatic place and route tool. Analysis tools evaluate the consistency or correctness of a design representation. The Cadence design rule checker (DRC) is an example of such a tool, it checks the physical layout for geometric rule violations. The Cadence electrical rule checker (ERC) is another example, it checks that the circuit extracted from the layout does not have any electrical errors such as shorts or floating nodes. Verification tools provide a formal method for demonstrating the equivalence of two design representations. The Cadence layout versus schematic (LVS) tool verifies that the circuit extracted from the layout is equivalent to the circuit design representation. Validation demonstrates the equivalence of two designs using a limited set of test cases. It is a less rigorous check than verification.

In the functional design stage the functional behavior of the design is represented. This may consist of timing charts, block diagrams, or behavioral models. Mentor Graphics supports models written in C, Pascal, FORTRAN, or VHDL. Behavioral simulation with a series of test vectors is used for analysis. Mentor Graphics Quicksim was used for behavioral simulation. Logic design involves implementing the functional design at the gate logic level using a schematic diagram. The logic design is validated against the functional design by comparing the results of the behavioral simulation with the logical simulation. Mentor Graphics Neted was used for schematic capture and Quicksim was once again used for logic simulation, using a unit gate delay. The circuit design phase involves implementing the logic design behavior with basic circuit elements such as transistors and capacitors. During the circuit design phase, transistors are sized to achieve drive and timing requirements and subsequently the logic design may undergo some minor changes as the gates are sized to match the load. Neted was used to make changes in the schematic and the Mentor Graphics Accusim SPICE simulator was used for timing analysis. During the physical design phase, the circuit design was converted into the geometric layout used in the fabrication process. The Mentor Graphics Cellgraph tool was used for automatic synthesis (placement and routing). To hand layout the standard cells and correct unrouted and poorly routed lines resulting from a Cellgraph automatic place and route, the designer may use Chipgraph to edit the physical layout. Cadence DRC, ERC, and LVS tools were used for analysis and verification. The layout is verified to conform to the design rules by the Cadence Dracula II design rule checker (DRC). The circuit extracted from the layout is checked for electrical errors such as shorts and floating nodes using its electrical rule checker (ERC), and verified to map to the schematic by its layout versus schematic (LVS) function. After the layout is completed, the designer can extract the layout capacitances, back annotate the

design, and re-simulate the circuit for greater accuracy using Cadence Dracula II layout parameter extraction (LPE).

Some of these design rule checking functions have recently been added by Mentor Graphics in their REMEDI and Checkmate programs. The Checkmate tool has been installed and setup by Chuck Carline, following the completion of the layout design work on this thesis. Checkmate can perform DRC, ERC, LVS, and LPE functions similar to the Cadence tool suite but it also has the same limitation of requiring users to exit from Mentor Graphics Chipgraph tool to run the checkers[12]. The advantage of the Checkmate installation over the Cadence installation is that the software licenses are not node locked to only two workstations, so they can run on any of the HP/Apollo workstations in the lab. At this time, Checkmate LPE has not yet been configured for the MOSIS 2.0 μm N-well CMOS process. The REMEDI tool is a design rule checker that Jeff Correll has integrated with the Chipgraph tools menus through a set of macros invoked at Chipgraph startup[13]. Because REMEDI is invoked from within the Chipgraph tool menus, it provides a more convenient first pass design rule check than the Cadence or Checkmate DRC tools, which must be invoked after exiting out from Chipgraph. On the other hand, REMEDI should only be used as a first pass DRC, because it does not use layout node data to skip same node checks or allow the definition of pseudo-layers. Therefore Cadence DRC or Checkmate should still be run on the final layout. When used in conjunction with Cadence or Checkmate DRC, REMEDI can reduce the number of invocations of Cadence or Checkmate DRC, and therefore the total layout design time.

A series of scripts developed by Shishir Ghate and Larry Rubin, designed to work in conjunction with the Mentor Graphics MCIF package, are used to convert the final Chipgraph layout data into the Caltech Intermediate Form (CIF) mask format accepted by MOSIS[1,2].

B. Tektronix Logic Verifier

The Tektronix Logic Verifier 500 (LV500) can be used to efficiently test complicated circuit designs. The system that is in current operation at RIT can handle up to 64 different bi-directional signals. It can be driven by up to four independent clocks (which can be distributed in any manner). The LV500 is primarily used for the testing of digital circuits and has been enhanced at RIT by the addition of a breadboard. With the breadboard an IC can be directly interfaced to the LV500. Another useful feature of the LV500 is its compatibility with the Mentor Graphics tools in RIT's VLSI Design Laboratory. Specifically the application TekWAVES can be used to translate Mentor Graphics test vector (event driven) format into the LV500's state driven format. Essentially TekWAVES samples the event state changes from the HP/Apollo state file and creates an output file that matches the state format and resolution of the LV500. Thus, one may design and test an IC using Mentor Graphics tools, fabricate that IC with the aid of RIT's Microelectronics facilities or MOSIS, and then test the circuit using the same test vectors that were run on the HP/Apollo workstations.

V. Method of Procedure

The overall procedure of this thesis followed these steps:

A. Write the Cadence Dracula II layout verification systems DRC, ERC, LVS, and LPE technology (".com") files for the MOSIS SCN 2.0 μm N-well CMOS process dense rules set in conjunction with Shishir Ghate and Larry Rubin.

B. Create additional library components and cells for the RIT CMOS standard cell library, as needed to complete the Morphological Array Processor Controller chip set.

C. Layout test chip 5 and 6 to test the functionality of the RIT CMOS standard cell library MUXes and D flip-flop cells.

D. Design a Morphological Array Processor Controller chip set using the RIT CMOS standard cell library.

1. The functional design architecture of the MAP Controller chip.

2. Perform the top level design of the MAP Controller chip set architecture, and partition the chip set.

3. Perform the logic design of the Controller and Memory Controller.

4. Simulate the logic design of the Controller and Memory Controller.

5. Perform the circuit design of the Controller and Memory Controller.

6. Simulate the circuit design of the Controller and Memory Controller.

7. Layout the Controller and Memory Controller.

8. Perform the design rule checking (DRC), electrical rule checking (ERC), and layout versus schematic (LVS) checking on the layouts.

9. Suggest a testing methodology.

A. Cadence Dracula II Layout Verification Tools Setup

In the design rule theory section, mention was made of the need for a separate technology definition file for each fabrication process and design rule checker; DRC, ERC, LVS, and LPE. Also mentioned in the theory section, was the need to define in the technology files the input layers and pseudo layers used by the rules checkers, and the specific design rules and parameter values for the fabrication process used. The MOSIS scaleable N-well CMOS (SCN) process dense rules set revision 6 was chosen to allow for more compacted designs.

The first step involved with defining the DRC technology file for the MOSIS SCN process (see Appendix A, DRC.COM File) was to define the input layers, the connection layers, and the connectivity relationship between the connect layers utilized by the process. The connect layer definitions are utilized for same node checking. The *INPUT-LAYER to *END block contains the input and connection layer definitions. The CONNECT x y BY z statements define the two layers to be connected, x and y, and the connection layer, z. The next step involved defining a minimum set of pseudo layers to define n and p transistor gates as well as other temporary layers used to perform various rules checks such as contact cut enclosure. A minimum number of layers is desirable, as each additional pseudo layer involves time and memory overhead. The AND, OR, XOR, and NOT statements are used to define the pseudo layers. The first two layers specify the input layers (possibly pseudo layers themselves) to perform the Boolean action upon, and the third layer the pseudo layer created. The final step involved defining a set of rules checks utilizing the input layers and pseudo layers to check for each of the dense rule set violations. A separate output error layer is specified to be created for each rule check violation so users will be aware of the design rule violated. WIDTH statements define minimum width checks, EXT statements define minimum spacing

checks, and ENC statements define enclosure checks. In the definitions the OUTPUT statement defines the output error cell layer. In most cases a single check is sufficient, but for those rules with multiple possible error combinations, such as rule 4.2 (pplus or nplus overlap of active), it was necessary to define an "a" and "b" set of rules checks to cover both possibilities. All of the rules checks were then verified using a test "chip" consisting of the specific examples from the MOSIS rev6 dense rule set documentation. In some cases additional pseudo layers were defined to minimize false errors and improve error detection, particularly the SELECT tests. In specifying the error checks, it is preferable to err on the side of generating some false errors rather than missing an actual error. Users can then use their best judgment to evaluate those errors flagged to determine if an error is a false one.

The ERC and LVS technology file definitions (see Appendix B, LVS.COM File) were able to borrow the input layer and pseudo layer definitions from the DRC definition file, after removal of those pseudo layers created specifically to check for DRC rule violations that are unnecessary for definition of the transistors within the layout. The next step for both rule checkers was to define the transistor nodes using the ELEMENT statement. For ERC it was also desirable to define the standard composite gate types, such as NAND and NOR gates, so output error reports would utilize that information to simplify and clarify output error reports, rather than reporting all output in terms of individual transistors alone.

The LPE technology file definition (see Appendix C, LPE.COM File) was similar to that for LVS, but also involved a step to define the pseudo layers that would be utilized for calculating parasitic capacitances and diodes, such as p diffusion to p substrate capacitance. The PARASITIC CAP statement defined these pseudo layers. A process specific value was then specified for each of

these capacitance types using the ATTRIBUTE CAP statement, based on data supplied by MOSIS. It was also specified that the capacitances should be "smashed" together into a single value for each node. As the tool did this after calculating each capacitance value separately, this turned out to be a significant limitation of the tool, as only 500,000 capacitance values could be evaluated prior to "smashing". This limited the usefulness of LPE to smaller layouts only.

B. Additional Standard Cells

In the course of completing the design for the MAP controller chip set it became necessary to design and layout additional RIT CMOS standard cell library components and cells. In particular additional D flip-flops with both Q and Q bar outputs, and-or-inverts, or-and-inverts, CMUXes, and MOSIS 64 and 84 pin pad rings were created (see Figure 5 through Figure 9). Each cell design was DRC, ERC, and LVS tested before addition to the RIT CMOS standard cell library.

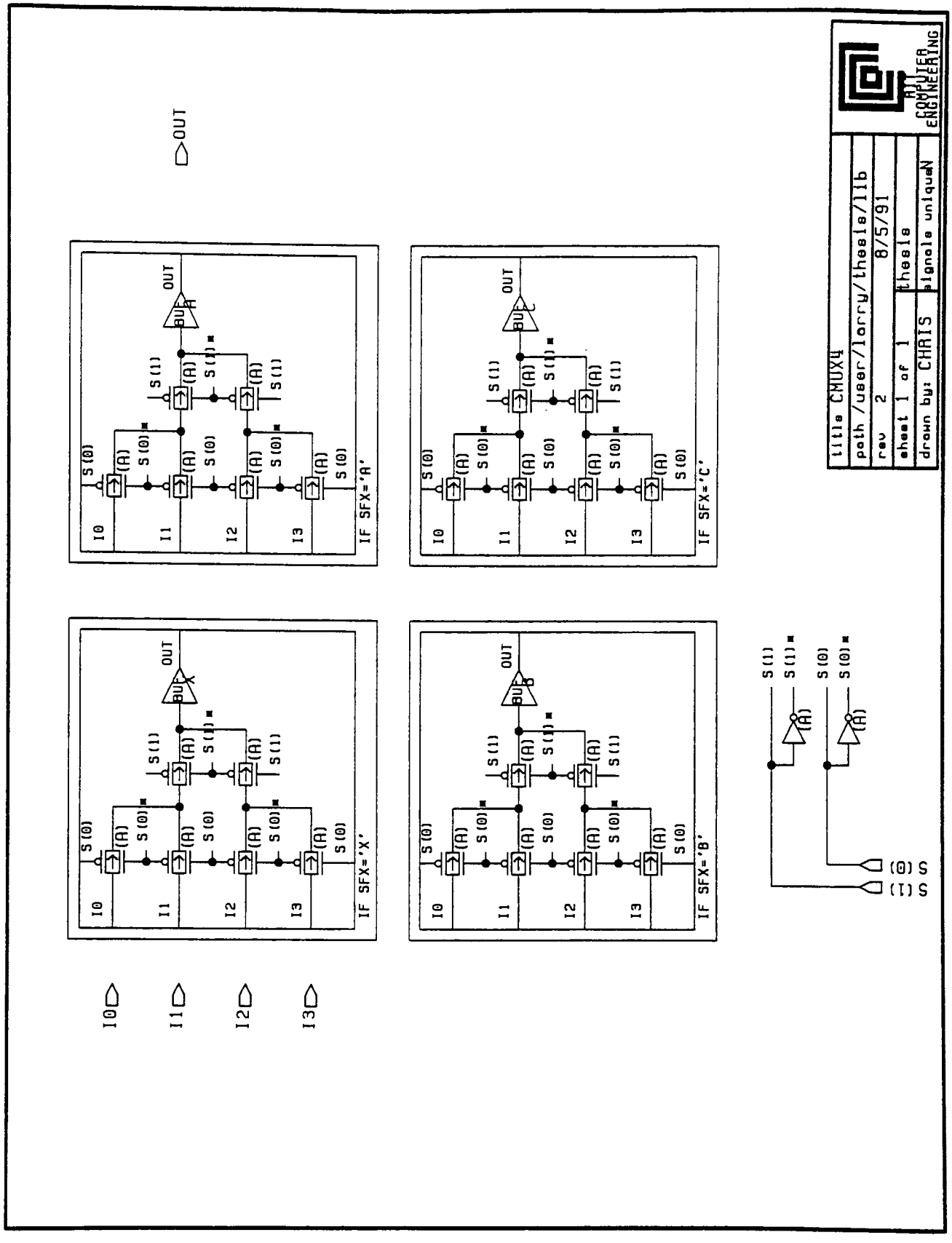

Figure 5 CMUX4 Schematic

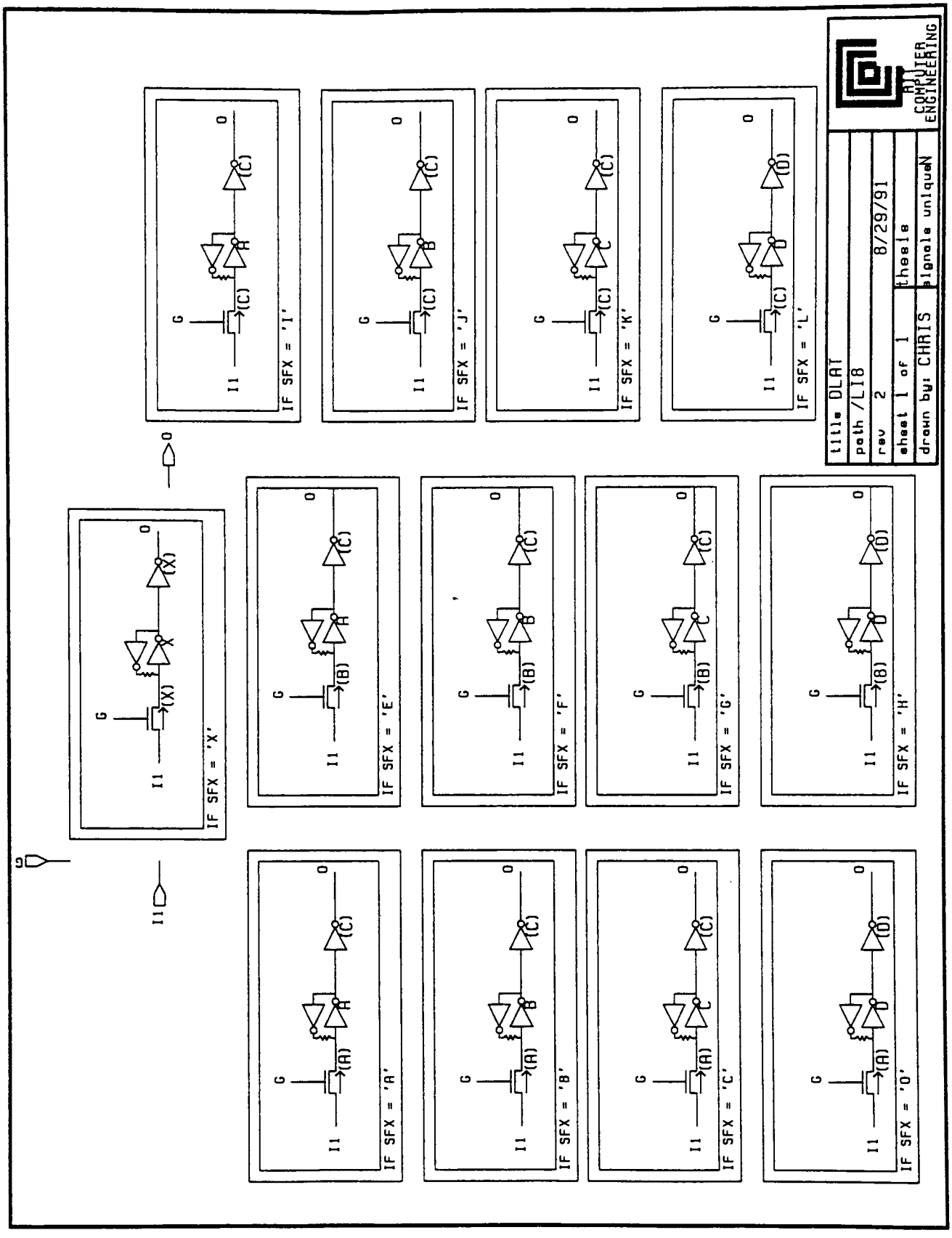

Figure 6 DLAT Schematic

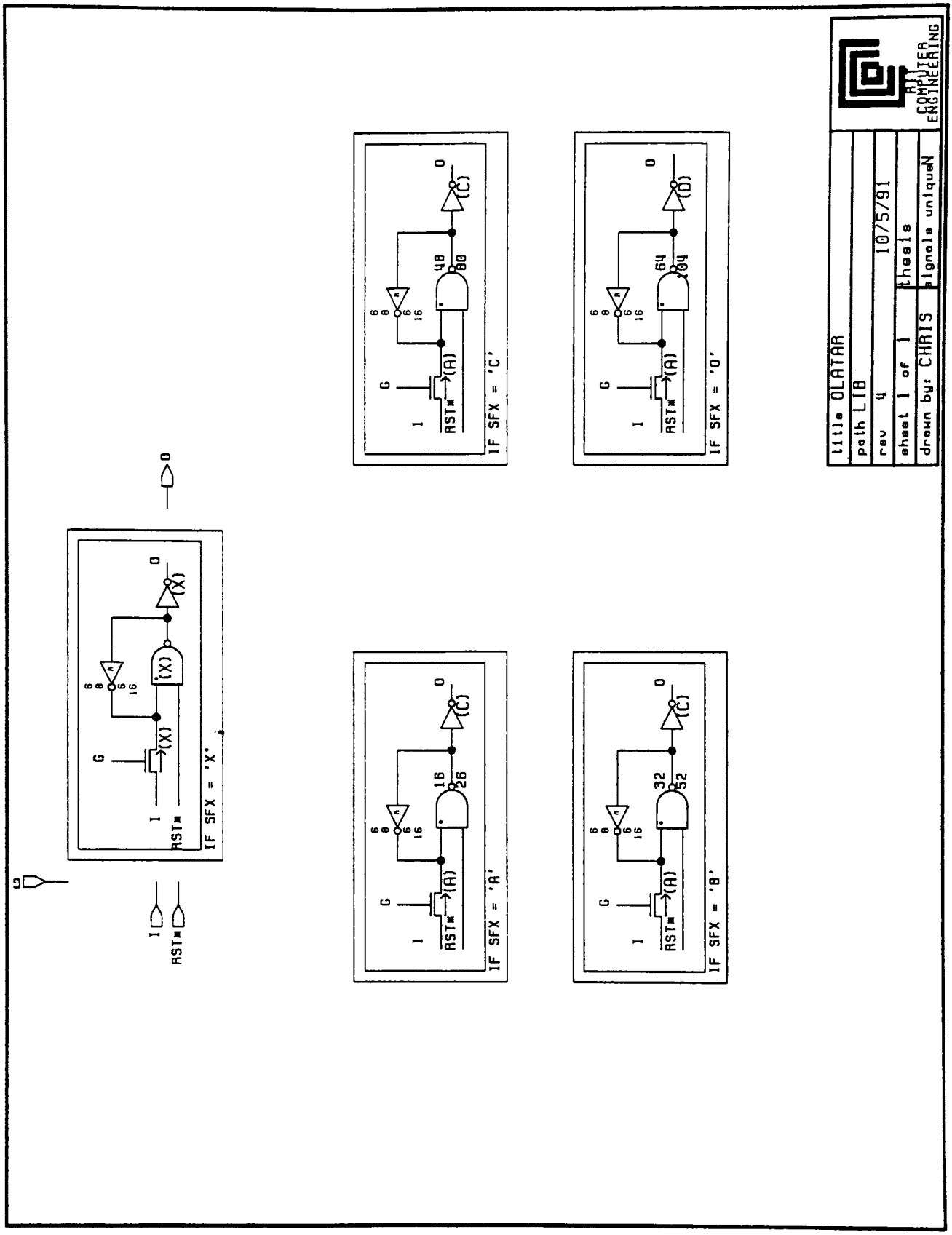

Figure 7 DLATAR Schematic

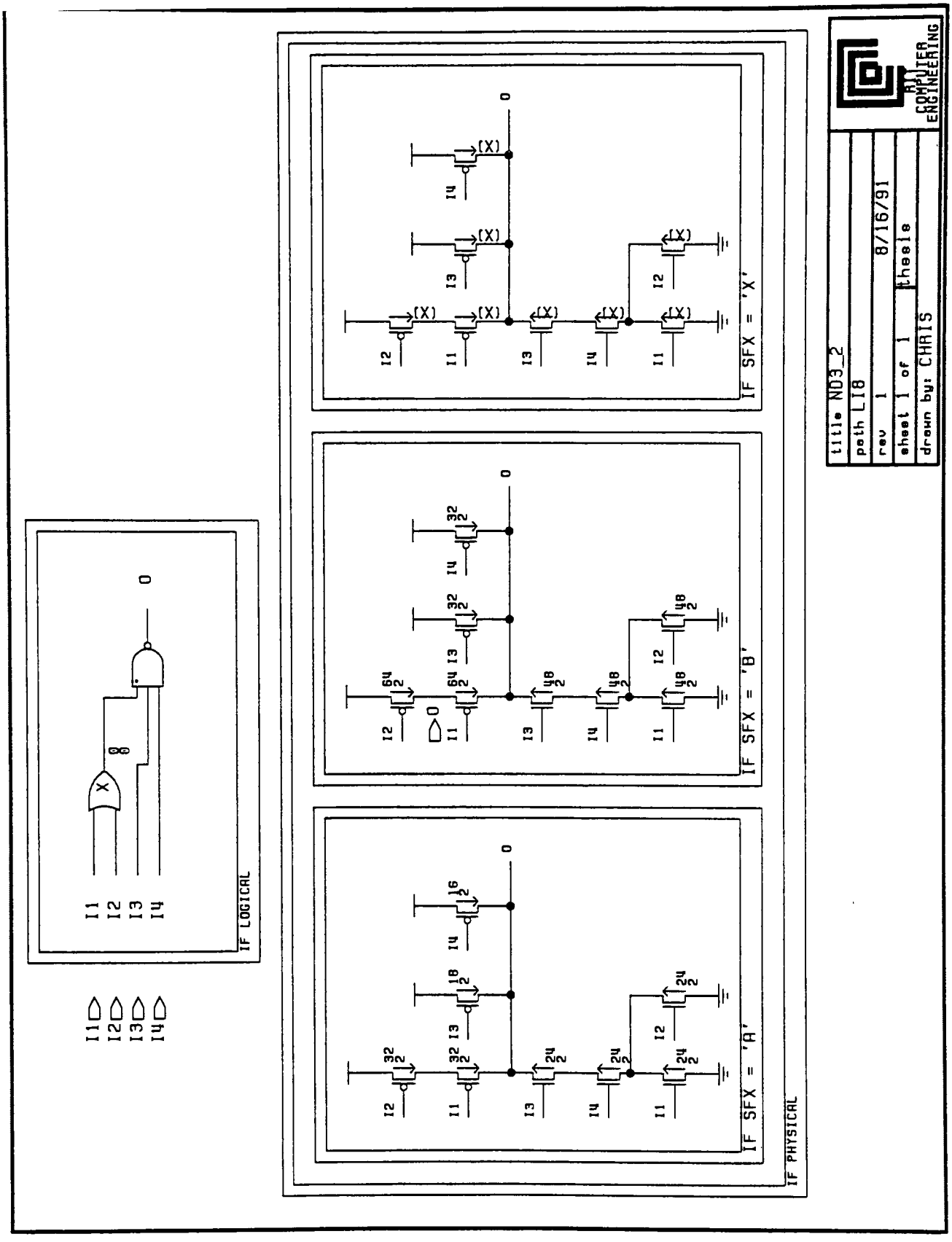

Figure 8 ND3_2 Schematic

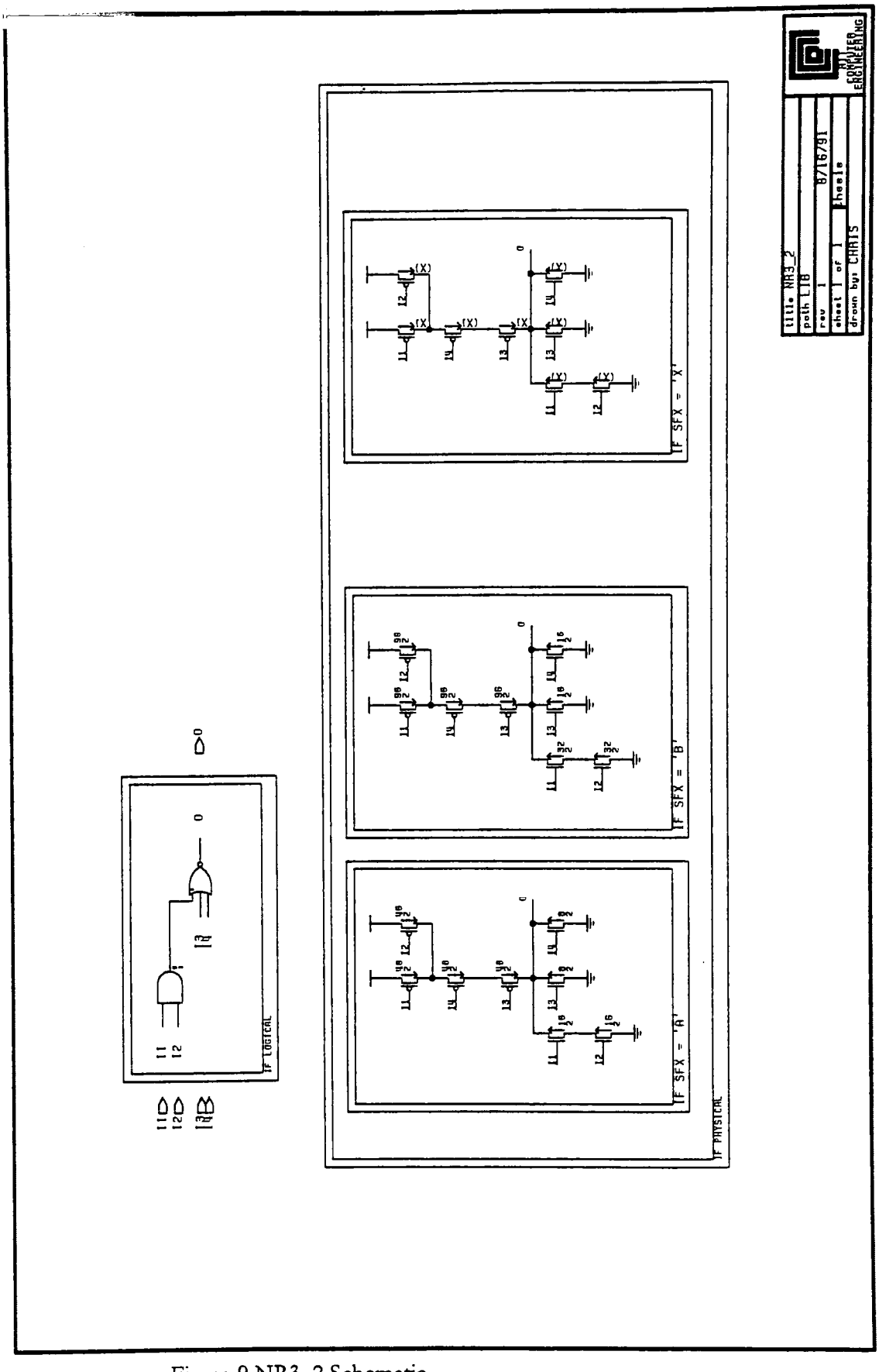

Figure 9 NR3_2 Schematic

C. Test Chips 5 and 6

As a part of his thesis, Shishir Ghate was asked to verify the RIT CMOS standard cell library designs and to calculate the internal node capacitances of the designs for use in back annotating the standard cell circuit schematics. Shishir performed the schematic design in Neted for the six test chips, along with the layout and verification of four of the test chips[2]. The layout and verification of the final two test chips was performed by the author. Test chip five contains size A through C two-input MUXes, size A and B three-input MUXes, size A four-input MUX, size A and B five-input MUXes, and a variety of D flip-flops (see Figure 10). Test chip six contains a variety of nMOS, pMOS, and CMOS transfer gates, individual p gate and n gate transistors, and a variety of inverters (see Figure 11). Each chip was layed out and verified using DRC, ERC, and LVS. Test chip four had an LVS error that the author was unable to locate prior to shipment of the test chip layouts to MOSIS for fabrication. This test chip contained some MUXes and D flip-flops whose design was later changed for other reasons, so the loss was not appreciable. The reason for the LVS errors turned out to be a result of the automatic place and route tool. Cellstation did not always complete the placement of every interconnection line during cell routing. The author only became aware of this problem later during cell place and route for the controller chip set, as the tools error message to this effect is difficult to find in the cell place and route transcript. Luckily the author was then aware of this issue and was able to verify the correct automatic place and route for both controller chips.

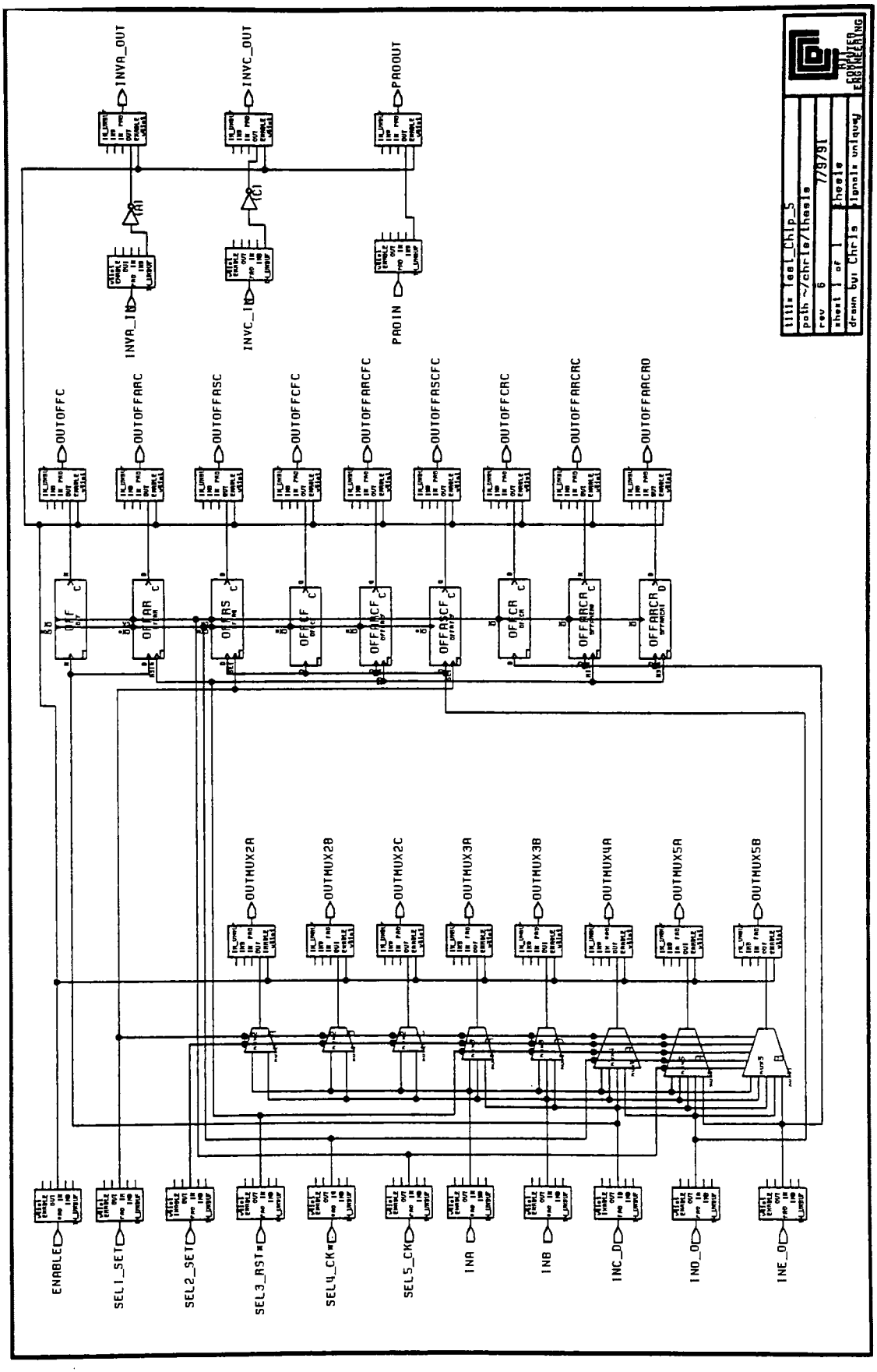

Figure 10 Test_Chip_5 Schematic

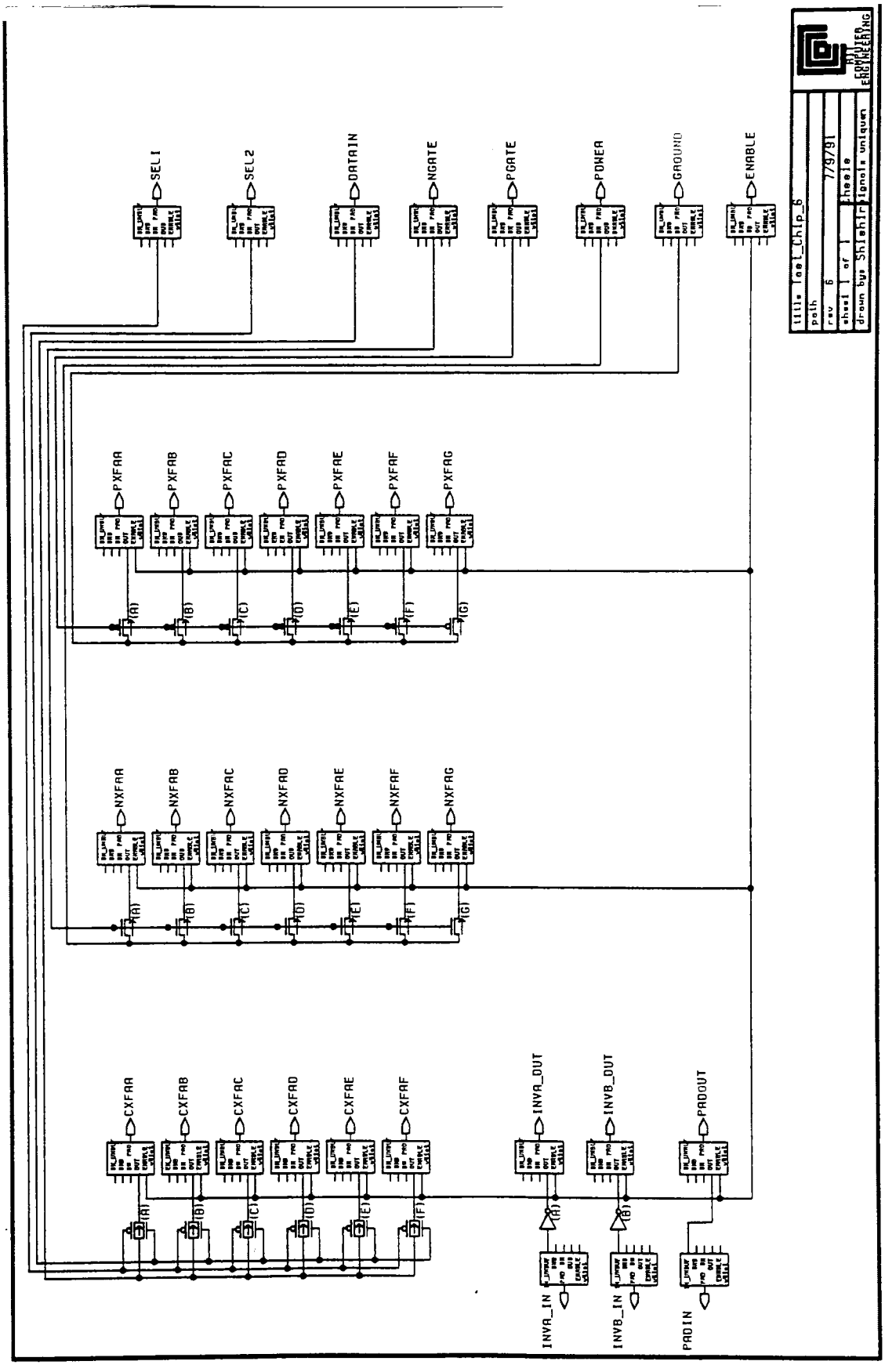

Figure 11 Test_Chip_6 Schematic

D. Morphological Array Processor Controller Design

1. The Functional Design Architecture of the MAP Controller Chip

The implementation of a morphological operation is accomplished by moving a window composed of patterning elements over an image. The target image pixel is the pixel under the center window element. Operations are performed on all pixels with a window value above them, with the resulting value replacing the original target pixel value.

When the construction of a Morphological Image Processor prototype was proposed, it was decided that the image would be 512x512 pixels using an 8-bit gray scale pixel representation, with an extra bit used to represent undefined pixels (shown as * in the Morphological Theory section). These undefined pixels can be implemented and treated as if they were negative infinity ($-\infty$) and will be represented as such in future references. The meaning of negative infinity was discussed in the Morphological Theory section and since it was possible to use the extra bit to also represent negative pixel values, they were included in case a use is ever found for them. Table 1 shows the binary pixel values and their corresponding decimal values. Negative values use the two's complement representation, with negative infinity using the value that would normally represent -512. Since most image capture and processing systems can only deal with 8-bit gray scale images and have no concept of negative infinity and negative pixel values, the 9-bit gray scale system will be referred to as an extended 8-bit gray scale system. The window was chosen to be a 7x7 array of

patterning elements, also using the extended 8-bit gray scale pixel representation.

Binary	Decimal
0 0000 0000	0
0 0000 0001	1
0 0000 0010	2
:	:
0 1111 1101	509
0 1111 1110	510
0 1111 1111	511
1 0000 0000	$-\infty$
1 0000 0001	-511
1 0000 0010	-510
:	:
1 1111 1101	-3
1 1111 1110	-2
1 1111 1111	-1

Table 1 Binary Pixel Values and Their Decimal Equivalents

While choosing an architecture for the Morphological Image Processor prototype, there were a few goals that were desirable to achieve. One of these goals was to allow the processor to operate at a real-time rate (60 images per second) using the same 512x512 pixel extended 8-bit gray scale images and a 7x7 window. Implementing the design using a VLSI circuit is currently the best way to achieve the real-time rate, so a regular structure was desirable to facilitate a VLSI layout. Another goal was a simple control section to facilitate expansion to larger window sizes and larger images. Also, the processor should be designed such that it is easy to pipeline with identical processors and allow the inputs and outputs to be connected to real-time sources and destinations, possibly with some buffering to make the system compatible with interleaved scan line systems.

For the Morphological Image Processor prototype, a 512x512 image and a 7x7 window were used, so all examples and explanations will use these sizes, unless otherwise noted. All of the concepts discussed in this section will work with any size image and window as long as appropriate adjustments are made. All index values will start at 0 and end at the one less than the maximum value, for example, the top left of a 512x512 image will be referred to as $X_{0,0}$ and the bottom right of that image will be referred to as $X_{511,511}$, with the first index value referring to the row and the second one to the column.

As previously discussed in the Morphological Theory section for equations (39) and (40), the morphological image processing operation is similar to convolution, with additions replacing the multiplications, and comparisons replacing the final additions. With a 512x512 image operated on by a 7x7 window, the values of every pixel in the image and its 48 neighbors are added to their corresponding window values (with the window centered over the pixel being operated on), and then either the maximum or minimum of those values, depending on the operation, is the resulting pixel value. The general mathematical equation follows:

$$Y_{i,j} = \text{compare}\left((X_{i+3,j+3} + W_{6,6}), (X_{i+3,j+2} + W_{6,5}), \ldots, (X_{i-3,j-3} + W_{0,0})\right) \qquad (43)$$

where the operands of the compare are 49 separate additions computed in parallel and the compare is either the maximum or minimum of the 49 results, depending on the desired morphological operation. In the equation, X is the input image, Y is the output image, W is the window matrix, i is the row index of the pixel being operated on (the target pixel), and j is the column index. A better indication of the 49 separate additions that are performed can be seen in Figure 12, where the top matrix denotes a portion of the image at any given time that will be added to the window matrix, which is shown on the bottom of the figure, with $X_{i,j}$ being the target pixel. The matrices are rotated 180° with

respect to conventional matrix representation to be consistent with future references. At the edges of an image, some of the values in the image matrix will not be valid (for example, when $X_{0,0}$ is the target pixel, all pixels to the bottom and to the right in the matrix will have negative indices making them invalid).

$X_{i+3,j+3}$	$X_{i+3,j+2}$	$X_{i+3,j+1}$	$X_{i+3,j}$	$X_{i+3,j-1}$	$X_{i+3,j-2}$	$X_{i+3,j-3}$
$X_{i+2,j+3}$	$X_{i+2,j+2}$	$X_{i+2,j+1}$	$X_{i+2,j}$	$X_{i+2,j-1}$	$X_{i+2,j-2}$	$X_{i+2,j-3}$
$X_{i+1,j+3}$	$X_{i+1,j+2}$	$X_{i+1,j+1}$	$X_{i+1,j}$	$X_{i+1,j-1}$	$X_{i+1,j-2}$	$X_{i+1,j-3}$
$X_{i,j+3}$	$X_{i,j+2}$	$X_{i,j+1}$	$X_{i,j}$	$X_{i,j-1}$	$X_{i,j-2}$	$X_{i,j-3}$
$X_{i-1,j+3}$	$X_{i-1,j+2}$	$X_{i-1,j+1}$	$X_{i-1,j}$	$X_{i-1,j-1}$	$X_{i-1,j-2}$	$X_{i-1,j-3}$
$X_{i-2,j+3}$	$X_{i-2,j+2}$	$X_{i-2,j+1}$	$X_{i-2,j}$	$X_{i-2,j-1}$	$X_{i-2,j-2}$	$X_{i-2,j-3}$
$X_{i-3,j+3}$	$X_{i-3,j+2}$	$X_{i-3,j+1}$	$X_{i-3,j}$	$X_{i-3,j-1}$	$X_{i-3,j-2}$	$X_{i-3,j-3}$

$W_{6,6}$	$W_{6,5}$	$W_{6,4}$	$W_{6,3}$	$W_{6,2}$	$W_{6,1}$	$W_{6,0}$
$W_{5,6}$	$W_{5,5}$	$W_{5,4}$	$W_{5,3}$	$W_{5,2}$	$W_{5,1}$	$W_{5,0}$
$W_{4,6}$	$W_{4,5}$	$W_{4,4}$	$W_{4,3}$	$W_{4,2}$	$W_{4,1}$	$W_{4,0}$
$W_{3,6}$	$W_{3,5}$	$W_{3,4}$	$W_{3,3}$	$W_{3,2}$	$W_{3,1}$	$W_{3,0}$
$W_{2,6}$	$W_{2,5}$	$W_{2,4}$	$W_{2,3}$	$W_{2,2}$	$W_{2,1}$	$W_{2,0}$
$W_{1,6}$	$W_{1,5}$	$W_{1,4}$	$W_{1,3}$	$W_{1,2}$	$W_{1,1}$	$W_{1,0}$
$W_{0,6}$	$W_{0,5}$	$W_{0,4}$	$W_{0,3}$	$W_{0,2}$	$W_{0,1}$	$W_{0,0}$

Figure 12 Image and Window Matrices

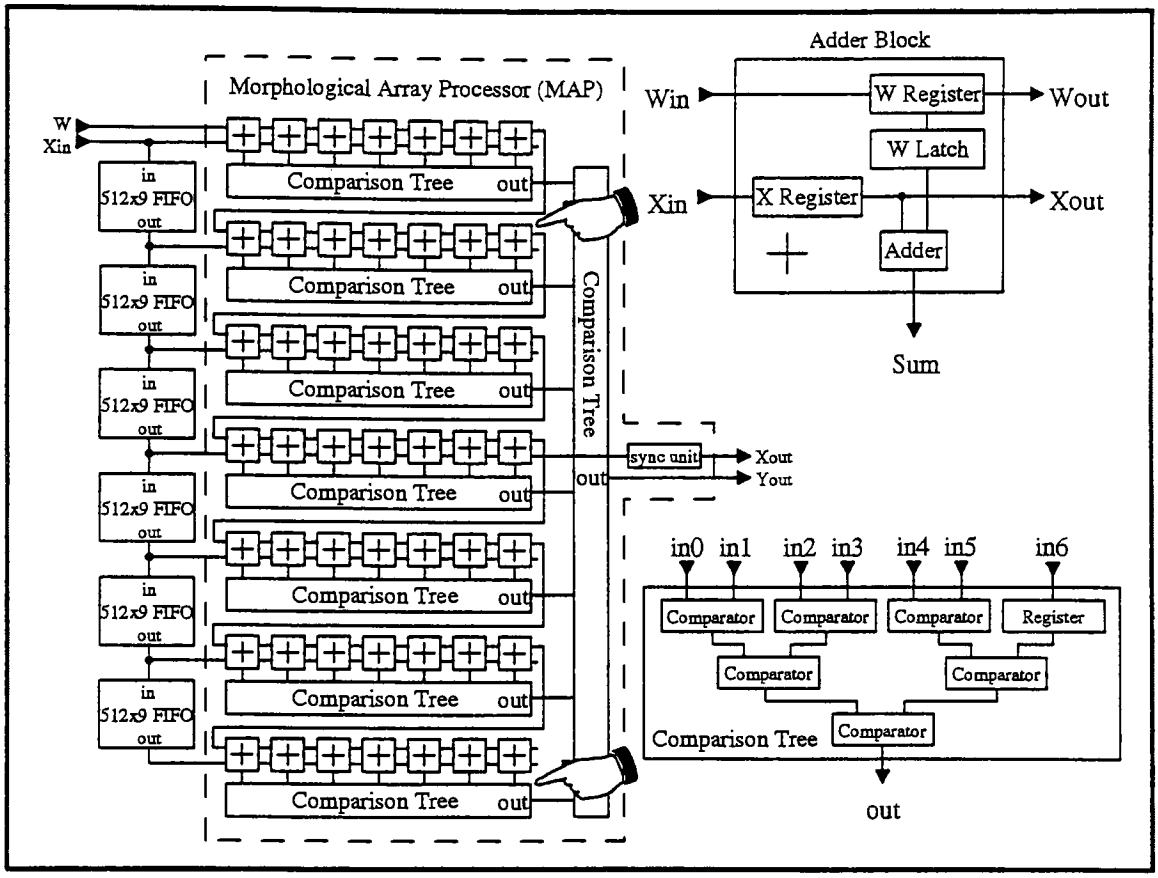

Figure 13 Morphological Array Processor Architecture

The architecture is suitable for pipelining, with the input format being the same as the output format. Figure 13 shows the general block diagram of the

architecture, which includes the MAP (Morphological Array Processor) and six external 512x9 bit FIFOs. Since this figure is meant to illustrate data flow, control signals have been omitted. X refers to image values, Y refers to the resulting image, and W refers to window values. Both the window array and the adder array are rotated 180° with respect to the normal visual orientation of an array for imaging operations, with the window value $W_{0,0}$ being in the lower right corner, $W_{0,6}$ in the lower left corner, $W_{6,0}$ in the upper right corner, and $W_{6,6}$ in the upper left corner, as shown on the bottom of Figure 12. This orientation is used so that the data can be shown going into the top left of the array and coming out of the bottom right. The FIFOs are external to the MAP, as that configuration will allow any image size to be used with the same MAP, as long as the right size FIFOs are used and the control circuitry is appropriately modified.

Before processing begins, the window elements are serially loaded into the 49 W registers contained in the adder blocks. The pixels of the first row of the 512x512 image are sent into the MAP from left to right at a rate of one pixel per clock cycle, immediately followed by the next row, also being sent in from left to right, continuing until all the rows have been sent into the MAP. The pixels are also simultaneously being sent into the 512 stage FIFOs, which are being used as line delays. Each of the FIFOs output image data that is exactly one image row above the output of the previous FIFO. For example, when FIFO #6 is sending the first row of image data into the last row of adder blocks, FIFO #5 is sending the second row of image data into the second to last row of adder blocks, FIFO #4 is sending the third row of image data into the fifth row of adder blocks, and so on. This will cause a 7x7 array of image data to be in the adder array at any given time, which will allow 49 simultaneous additions with the window patterning elements.

At any given time during processing, the pixel that is being operated on (the target pixel) is in the middle of the 7x7 array of adder blocks. All pixel values in the array are added to the window values residing in the same adder block with the exception of pixels that are blanked. Blanking occurs when the edges of the image are being operated on, and unwanted pixels are in the 7x7 array, such as pixels from the opposite edge, pixels from the previous image, pixels from the next image, or unknown values when no image is directly preceding or following the image being processed. When a row and/or column is blanked, that row and/or column is not used for calculating the new target pixel value. For example, row blanking invalidates pixels from the right edge of the image when the left edge is being processed, and column blanking invalidates pixels from a previous image (or no image at all) when the first few rows of an image are being processed.

Figure 15 through Figure 26 show how the blanking is implemented, with the figures showing 5x5 arrays and the image values contained in the corresponding adder blocks. 5x5 arrays have been used in place of 7x7 arrays for space considerations. The first and last rows and the first and last columns of a 7x7 array have been removed to create a 5x5 array, and the difference in the blanking procedure will be discussed later. Each figure shows six consecutive cycles which surround a critical moment in the blanking process. Shaded rows and/or columns indicate which rows and/or columns are being blanked and the values in bold face indicates the target pixels. Any pixel value with a *prev* or *next* indicates a pixel from the previous or next image, respectively. Whenever a row and/or column needs to be blanked, the whole row and/or column must be blanked, as all of the pixels in that row and/or column are not valid for operating on the target pixel. In the example figures, each image is followed by another image without any delay, so it can be seen that in a real-time application no delay between images is necessary. If desired, any delay greater

than zero cycles can also be used, as any value that is not in the current image is always blanked.

To implement the blanking, control bits are needed to instruct each row and each column whether to blank or not. When a row and/or a column is blanked, the output of every adder in that row and/or column is set to negative infinity. If a morphological erosion (minimum) is being performed, the negative infinity will propagate to the output, which is the desired effect for erosion. For a dilation (maximum), the negative infinity will have no effect on the output as negative infinity is the smallest defined value, which is also the desired effect. For a 7x7 window, no more than three rows and three columns can be blanked at any time, and the center row and center column of the adder array can never be blanked because the middle element of the array always contains the target pixel.

Figure 14 shows the bit numbers that will be used for both 5x5 and 7x7 windows to refer to the rows and/or columns that are to be blanked. Rows/columns 0 and 5 (the outer rows and columns) have been removed from the 7x7 array to make the 5x5 array. The elements that are missing from a 5x5 array that would make up a 7x7 can be determined by referencing the array on the top of Figure 12.

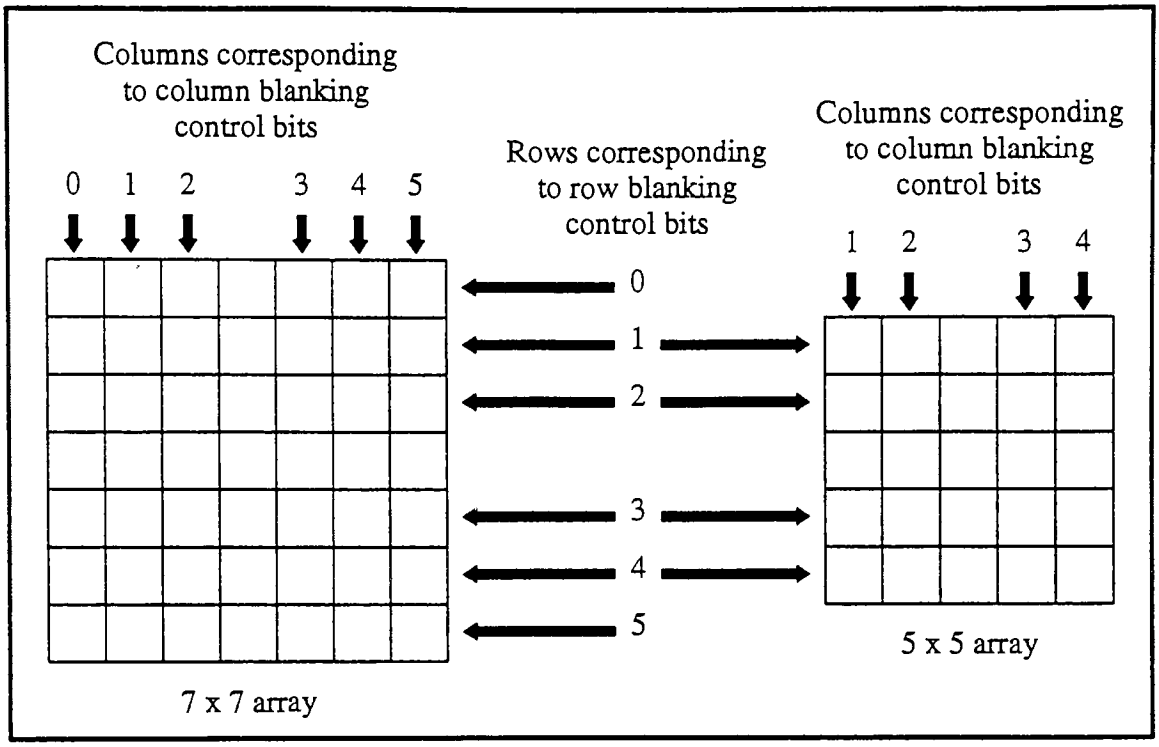

Figure 14 Blanking Bits

ROW OR COLUMN INDEX	7 X 7 BLANKING CONTROL BITS						5 X 5 BLANKING CONTROL BITS			
	0	1	2	3	4	5	1	2	3	4
0	0	0	0	1	1	1	0	0	1	1
1	0	0	0	0	1	1	0	0	0	1
2	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
:	0	0	0	0	0	0	0	0	0	0
508	0	0	0	0	0	0	0	0	0	0
509	1	0	0	0	0	0	0	0	0	0
510	1	1	0	0	0	0	1	0	0	0
511	1	1	1	0	0	0	1	1	0	0

Table 2 Blanking Control Values For All Index Values

Table 2 shows what the blanking control bits will be for all row and column index values of the target pixel. For a 7x7 window operating on a 512x512 image, no row blanking occurs between target pixel row index values of 3 and 508, as the rows in the adder array are far enough into the image to contain only valid pixels. Likewise, no column blanking occurs between target pixel column index values 3 and 508, as the pixels from the opposite edge (when near either the left or right edge) are no longer in the adder array. As can be seen in the table, a 7x7 window needs one more blanking control bit for each edge than a 5x5 window does (bit 0 on the top and left edges and bit 5 on the bottom and right edges). Each one of those four extra control bits (two for row blanking and two for column blanking) will be blanked one more cycle than its neighboring bit. For example, when the target pixel is $X_{0,0}$, $X_{0,1}$, or $X_{0,2}$, column blanking control bit 5 will specify that its corresponding column should be blanked.

TARGET PIXEL (ROW,COLUMN)	ROW BLANKING BITS						COLUMN BLANKING BITS					
	0	1	2	3	4	5	0	1	2	3	4	5
0,0	0	0	0	1	1	1	0	0	0	1	1	1
0,1	0	0	0	1	1	1	0	0	0	0	1	1
0,2	0	0	0	1	1	1	0	0	0	0	0	1
0,3	0	0	0	1	1	1	0	0	0	0	0	0
:	0	0	0	1	1	1	0	0	0	0	0	0
0,508	0	0	0	1	1	1	0	0	0	0	0	0
0,509	0	0	0	1	1	1	1	0	0	0	0	0
0,510	0	0	0	1	1	1	1	1	0	0	0	0
0,511	0	0	0	1	1	1	1	1	1	0	0	0
1,0	0	0	0	0	1	1	0	0	0	1	1	1
1,1	0	0	0	0	1	1	0	0	0	0	1	1
1,2	0	0	0	0	1	1	0	0	0	0	0	1
1,3	0	0	0	0	1	1	0	0	0	0	0	0
:	0	0	0	0	1	1	0	0	0	0	0	0
1,508	0	0	0	0	1	1	0	0	0	0	0	0
1,509	0	0	0	0	1	1	1	0	0	0	0	0
1,510	0	0	0	0	1	1	1	1	0	0	0	0
1,511	0	0	0	0	1	1	1	1	1	0	0	0
2,0	0	0	0	0	0	1	0	0	0	1	1	1
2,1	0	0	0	0	0	1	0	0	0	0	1	1
2,2	0	0	0	0	0	1	0	0	0	0	0	1
2,3	0	0	0	0	0	1	0	0	0	0	0	0
:	0	0	0	0	0	1	0	0	0	0	0	0
2,508	0	0	0	0	0	1	0	0	0	0	0	0
2,509	0	0	0	0	0	1	1	0	0	0	0	0
2,510	0	0	0	0	0	1	1	1	0	0	0	0
2,511	0	0	0	0	0	1	1	1	1	0	0	0
3,0	0	0	0	0	0	0	0	0	0	1	1	1
3,1	0	0	0	0	0	0	0	0	0	0	1	1
3,2	0	0	0	0	0	0	0	0	0	0	0	1
3,3	0	0	0	0	0	0	0	0	0	0	0	0

Table 3 Blanking Control Values

TARGET PIXEL (ROW,COLUMN)	ROW BLANKING BITS						COLUMN BLANKING BITS					
	0	1	2	3	4	5	0	1	2	3	4	5
508,508	0	0	0	0	0	0	0	0	0	0	0	0
508,509	0	0	0	0	0	0	1	0	0	0	0	0
508,510	0	0	0	0	0	0	1	1	0	0	0	0
508,511	0	0	0	0	0	0	1	1	1	0	0	0
509,0	1	0	0	0	0	0	0	0	0	1	1	1
509,1	1	0	0	0	0	0	0	0	0	0	1	1
509,2	1	0	0	0	0	0	0	0	0	0	0	1
509,3	1	0	0	0	0	0	0	0	0	0	0	0
:	1	0	0	0	0	0	0	0	0	0	0	0
509,508	1	0	0	0	0	0	0	0	0	0	0	0
509,509	1	0	0	0	0	0	1	0	0	0	0	0
509,510	1	0	0	0	0	0	1	1	0	0	0	0
509,511	1	0	0	0	0	0	1	1	1	0	0	0
510,0	1	1	0	0	0	0	0	0	0	1	1	1
510,1	1	1	0	0	0	0	0	0	0	0	1	1
510,2	1	1	0	0	0	0	0	0	0	0	0	1
510,3	1	1	0	0	0	0	0	0	0	0	0	0
:	1	1	0	0	0	0	0	0	0	0	0	0
510,508	1	1	0	0	0	0	0	0	0	0	0	0
510,509	1	1	0	0	0	0	1	0	0	0	0	0
510,510	1	1	0	0	0	0	1	1	0	0	0	0
510,511	1	1	0	0	0	0	1	1	1	0	0	0
511,0	1	1	1	0	0	0	0	0	0	1	1	1
511,1	1	1	1	0	0	0	0	0	0	0	1	1
511,2	1	1	1	0	0	0	0	0	0	0	0	1
511,3	1	1	1	0	0	0	0	0	0	0	0	0
:	1	1	1	0	0	0	0	0	0	0	0	0
511,508	1	1	1	0	0	0	0	0	0	0	0	0
511,509	1	1	1	0	0	0	1	0	0	0	0	0
511,510	1	1	1	0	0	0	1	1	0	0	0	0
511,511	1	1	1	0	0	0	1	1	1	0	0	0

Table 4 Blanking Control Values (cont'd)

Table 3 and Table 4 show more complete row and column blanking control bit values for a 7x7 window, with most of the critical areas of blanking directly corresponding to the 5x5 examples in Figure 15 through Figure 26. In the gaps between the left side and the right side of the image, no column blanking takes place and row blanking only occurs at the upper and lower edges of the image. In the gap between the upper and lower edges of the image, no row blanking takes place and column blanking continues as usual at the sides of the image.

1,511	1,510	1,509	1,508	1,507
0,511	0,510	0,509	0,508	0,507
prev 511,511	prev 511,510	**prev 511,509**	prev 511,508	prev 511,507
prev 510,511	prev 510,510	prev 510,509	prev 510,508	prev 510,507
prev 509,511	prev 509,510	prev 509,509	prev 509,508	prev 509,507

2,0	1,511	1,510	1,509	1,508
1,0	0,511	0,510	0,509	0,508
prev 0,0	prev 511,511	**prev 511,510**	prev 511,509	prev 511,508
prev 511,0	prev 510,511	prev 510,510	prev 510,509	prev 510,508
prev 510,0	prev 509,511	prev 509,510	prev 509,509	prev 509,508

2,1	2,0	1,511	1,510	1,509
1,1	1,0	0,511	0,510	0,509
0,1	0,0	**prev 511,511**	prev 511,510	prev 511,509
prev 511,1	prev 511,0	prev 510,511	prev 510,510	prev 510,509
prev 510,1	prev 510,0	prev 509,511	prev 509,510	prev 509,509

Figure 15 Blanking part 1

2,2	2,1	2,0	1,511	1,510
1,2	1,1	1,0	0,511	0,510
0,2	0,1	**0,0**	prev 511,511	prev 511,510
prev 511,2	prev 511,1	prev 511,0	prev 510,511	prev 510,510
prev 510,2	prev 510,1	prev 510,0	prev 509,511	prev 509,510

2,3	2,2	2,1	2,0	1,511
1,3	1,2	1,1	1,0	0,511
0,3	0,2	**0,1**	0,0	prev 511,511
prev 511,3	prev 511,2	prev 511,1	prev 511,0	prev 510,511
prev 510,3	prev 510,2	prev 510,1	prev 510,0	prev 509,511

2,4	2,3	2,2	2,1	2,0
1,4	1,3	1,2	1,1	1,0
0,4	0,3	**0,2**	0,1	0,0
prev 511,4	prev 511,3	prev 511,2	prev 511,1	prev 511,0
prev 510,4	prev 510,3	prev 510,2	prev 510,1	prev 510,0

Figure 16 Blanking part 2

2,511	2,510	2,509	2,508	2,507
1,511	1,510	1,509	1,508	1,507
0,511	0,510	**0,509**	0,508	0,507
prev 511,511	prev 511,510	prev 511,509	prev 511,508	prev 511,507
prev 510,511	prev 510,510	prev 510,509	prev 510,508	prev 510,507

3,0	2,511	2,510	2,509	2,508
2,0	1,511	1,510	1,509	1,508
1,0	0,511	**0,510**	0,509	0,508
0,0	prev 511,511	prev 511,510	prev 511,509	prev 511,508
prev 511,0	prev 510,511	prev 510,510	prev 510,509	prev 510,508

3,1	3,0	2,511	2,510	2,509
2,1	2,0	1,511	1,510	1,509
1,1	1,0	**0,511**	0,510	0,509
0,1	0,0	prev 511,511	prev 511,510	prev 511,509
prev 511,1	prev 511,0	prev 510,511	prev 510,510	prev 510,509

Figure 17 Blanking part 3

3,2	3,1	3,0	2,511	2,510
2,2	2,1	2,0	1,511	1,510
1,2	1,1	**1,0**	0,511	0,510
0,2	0,1	0,0	prev 511,511	prev 511,510
prev 511,2	prev 511,1	prev 511,0	prev 510,511	prev 510,510

3,3	3,2	3,1	3,0	2,511
2,3	2,2	2,1	2,0	1,511
1,3	1,2	**1,1**	1,0	0,511
0,3	0,2	0,1	0,0	prev 511,511
prev 511,3	prev 511,2	prev 511,1	prev 511,0	prev 510,511

3,4	3,3	3,2	3,1	3,0
2,4	2,3	2,2	2,1	2,0
1,4	1,3	**1,2**	1,1	1,0
0,4	0,3	0,2	0,1	0,0
prev 511,4	prev 511,3	prev 511,2	prev 511,1	prev 511,0

Figure 18 Blanking part 4

3,511	3,510	3,509	3,508	3,507
2,511	2,510	2,509	2,508	2,507
1,511	1,510	**1,509**	1,508	1,507
0,511	0,510	0,509	0,508	0,507
prev 511,511	prev 511,510	prev 511,509	prev 511,508	prev 511,507

4,0	3,511	3,510	3,509	3,508
3,0	2,511	2,510	2,509	2,508
2,0	1,511	**1,510**	1,509	1,508
1,0	0,511	0,510	0,509	0,508
0,0	prev 511,511	prev 511,510	prev 511,509	prev 511,508

4,1	4,0	3,511	3,510	3,509
3,1	3,0	2,511	2,510	2,509
2,1	2,0	**1,511**	1,510	1,509
1,1	1,0	0,511	0,510	0,509
0,1	0,0	prev 511,511	prev 511,510	prev 511,509

Figure 19 Blanking part 5

4,2	4,1	4,0	3,511	3,510
3,2	3,1	3,0	2,511	2,510
2,2	2,1	**2,0**	1,511	1,510
1,2	1,1	1,0	0,511	0,510
0,2	0,1	0,0	prev 511,511	prev 511,510

4,3	4,2	4,1	4,0	3,511
3,3	3,2	3,1	3,0	2,511
2,3	2,2	**2,1**	2,0	1,511
1,3	1,2	1,1	1,0	0,511
0,3	0,2	0,1	0,0	prev 511,511

4,4	4,3	4,2	4,1	4,0
3,4	3,3	3,2	3,1	3,0
2,4	2,3	**2,2**	2,1	2,0
1,4	1,3	1,2	1,1	1,0
0,4	0,3	0,2	0,1	0,0

Figure 20 Blanking part 6

511,511	511,510	511,509	511,508	511,507
510,511	510,510	510,509	510,508	510,507
509,511	509,510	**509,509**	509,508	509,507
508,511	508,510	508,509	508,508	508,507
507,511	507,510	507,509	507,508	507,507

next 0,0	511,511	511,510	511,509	511,508
511,0	510,511	510,510	510,509	510,508
510,0	509,511	**509,510**	509,509	509,508
509,0	508,511	508,510	508,509	508,508
508,0	507,511	507,510	507,509	507,508

next 0,1	next 0,0	511,511	511,510	511,509
511,1	511,0	510,511	510,510	510,509
510,1	510,0	**509,511**	509,510	509,509
509,1	509,0	508,511	508,510	508,509
508,1	508,0	507,511	507,510	507,509

Figure 21 Blanking part 7

next 0,2	next 0,1	next 0,0	511,511	511,510
511,2	511,1	511,0	510,511	510,510
510,2	510,1	**510,0**	509,511	509,510
509,2	509,1	509,0	508,511	508,510
508,2	508,1	508,0	507,511	507,510

next 0,3	next 0,2	next 0,1	next 0,0	511,511
511,3	511,2	511,1	511,0	510,511
510,3	510,2	**510,1**	510,0	509,511
509,3	509,2	509,1	509,0	508,511
508,3	508,2	508,1	508,0	507,511

next 0,4	next 0,3	next 0,2	next 0,1	next 0,0
511,4	511,3	511,2	511,1	511,0
510,4	510,3	**510,2**	510,1	510,0
509,4	509,3	509,2	509,1	509,0
508,4	508,3	508,2	508,1	508,0

Figure 22 Blanking part 8

next 0,511	next 0,510	next 0,509	next 0,508	next 0,507
511,511	511,510	511,509	511,508	511,507
510,511	510,510	**510,509**	510,508	510,507
509,511	509,510	509,509	509,508	509,507
508,511	508,510	508,509	508,508	508,507

next 1,0	next 0,511	next 0,510	next 0,509	next 0,508
next 0,0	511,511	511,510	511,509	511,508
511,0	510,511	**510,510**	510,509	510,508
510,0	509,511	509,510	509,509	509,508
509,0	508,511	508,510	508,509	508,508

next 1,1	next 1,0	next 0,511	next 0,510	next 0,509
next 0,1	next 0,0	511,511	511,510	511,509
511,1	511,0	**510,511**	510,510	510,509
510,1	510,0	509,511	509,510	509,509
509,1	509,0	508,511	508,510	508,509

Figure 23 Blanking part 9

next 1,2	next 1,1	next 1,0	next 0,511	next 0,510
next 0,2	next 0,1	next 0,0	511,511	511,510
511,2	511,1	**511,0**	510,511	510,510
510,2	510,1	510,0	509,511	509,510
509,2	509,1	509,0	508,511	508,510

next 1,3	next 1,2	next 1,1	next 1,0	next 0,511
next 0,3	next 0,2	next 0,1	next 0,0	511,511
511,3	511,2	**511,1**	511,0	510,511
510,3	510,2	510,1	510,0	509,511
509,3	509,2	509,1	509,0	508,511

next 1,4	next 1,3	next 1,2	next 1,1	next 1,0
next 0,4	next 0,3	next 0,2	next 0,1	next 0,0
511,4	511,3	**511,2**	511,1	511,0
510,4	510,3	510,2	510,1	510,0
509,4	509,3	509,2	509,1	509,0

Figure 24 Blanking part 10

next 1,511	next 1,510	next 1,509	next 1,508	next 1,507
next 0,511	next 0,510	next 0,509	next 0,508	next 0,507
511,511	511,510	**511,509**	511,508	511,507
510,511	510,510	510,509	510,508	510,507
509,511	509,510	509,509	509,508	509,507

next 2,0	next 1,511	next 1,510	next 1,509	next 1,508
next 1,0	next 0,511	next 0,510	next 0,509	next 0,508
next 0,0	511,511	**511,510**	511,509	511,508
511,0	510,511	510,510	510,509	510,508
510,0	509,511	509,510	509,509	509,508

next 2,1	next 2,0	next 1,511	next 1,510	next 1,509
next 1,1	next 1,0	next 0,511	next 0,510	next 0,509
next 0,1	next 0,0	**511,511**	511,510	511,509
511,1	511,0	510,511	510,510	510,509
510,1	510,0	509,511	509,510	509,509

Figure 25 Blanking part 11

next 2,2	next 2,1	next 2,0	next 1,511	next 1,510
next 1,2	next 1,1	next 1,0	next 0,511	next 0,510
next 0,2	next 0,1	**next 0,0**	511,511	511,510
511,2	511,1	511,0	510,511	510,510
510,2	510,1	510,0	509,511	509,510

next 2,3	next 2,2	next 2,1	next 2,0	next 1,511
next 1,3	next 1,2	next 1,1	next 1,0	next 0,511
next 0,3	next 0,2	**next 0,1**	next 0,0	511,511
511,3	511,2	511,1	511,0	510,511
510,3	510,2	510,1	510,0	509,511

next 2,4	next 2,3	next 2,2	next 2,1	next 2,0
next 1,4	next 1,3	next 1,2	next 1,1	next 1,0
next 0,4	next 0,3	**next 0,2**	next 0,1	next 0,0
511,4	511,3	511,2	511,1	511,0
510,4	510,3	510,2	510,1	510,0

Figure 26 Blanking part 12

After all the additions are performed in parallel, there are 49 results waiting to be compared to each other to determine which value is the maximum or the minimum, depending on the operation. Since it is impractical to have a 49-input comparator, a comparison tree composed of 2-input comparators and synchronization registers is a much more feasible solution. See Figure 13 for the general block diagram of the comparison tree. Each comparison stage consists of a comparator and an output holding register, with all of the output holding registers tied to a common clock, so it will take a minimum of 6 clock cycles to complete all of the comparisons needed to generate an output pixel. Since a comparison tree is being used and all comparators are properly clocked and synchronized, the tree will output one resulting pixel value per clock cycle. Each level of the comparison tree contains intermediate results for successive pixels. Further references to intermediate results of an operation on any input pixel $X_{i,j}$ will be denoted as $I_{i,j}$, with $I_{i,j}$ having the possibility of referring to numerous intermediate results.

For example, when the first pixel of the image ($X_{0,0}$) is in the target pixel position (the middle of the adder array), all 49 additions for that pixel are performed simultaneously. On the next clock cycle, the 49 results are sent into the first level of the comparison tree and the results are ready before the end of the clock cycle. Meanwhile, the second pixel of the image ($X_{0,1}$) is in the target position and the 49 additions for that pixel are taking place. On the next cycle, the intermediate results of $X_{0,0}$ ($I_{0,0}$) are sent into the second level of the comparison tree, and $I_{0,1}$ is sent into the first level, and $X_{0,2}$ is now the target pixel. This will continue until $I_{0,0}$ is in the sixth level of the tree and $X_{0,6}$ is the target pixel. On the next cycle, the output of the final comparator in the tree will be the first output pixel of the morphological array processor, $Y_{0,0}$. The proposed tree in Figure 13 is only one possible way to implement it, many

other arrangements can be used as long as the intermediate results remain synchronized.

2. The Top Level Design of the MAP Controller Chip Set

The controller chip set controls most of the other chips in the Morphological Array Image Processor (see Figure 27). Processing instructions from the host computer are written to registers within the Controller, and these instructions are then used to generate additional control signals to control the Mem_Control chips, the MAP, and both ALUs. The Morphological Array Image Processor consists of four memories each capable of containing one image. To provide control for these memories, each memory has an associated memory control (Mem_Control). The Mem_Control accept control signals from the Controller, and the host bus interface including the PC bus address interface. The control and address signals from the bus interface allow the host computer to access on-board image memories, and to perform image processing operations using images stored in the on-board memories. Whenever the host computer needs to read from, or write to, the on-board memories it issues the appropriate signals to the bus interface. The bus interface generates the signals to enable the buffer, a PC chip select signal for every Mem_Control, a read/write signal used by all Mem_Controls, and the appropriate PC address. An enabled Mem_Control PC chip select signal will select the PC address as opposed to the address internally generated within the Mem_Control. The Mem_Control chip is also used for image processing operations. One or two memories are read from to provide input data, and the output is written to another memory. The Controller chip informs each Mem_Control whether it will be used for reading or writing and when to start the address counter. Each Mem_Control has a *START_MEM* input to reset the address counter.

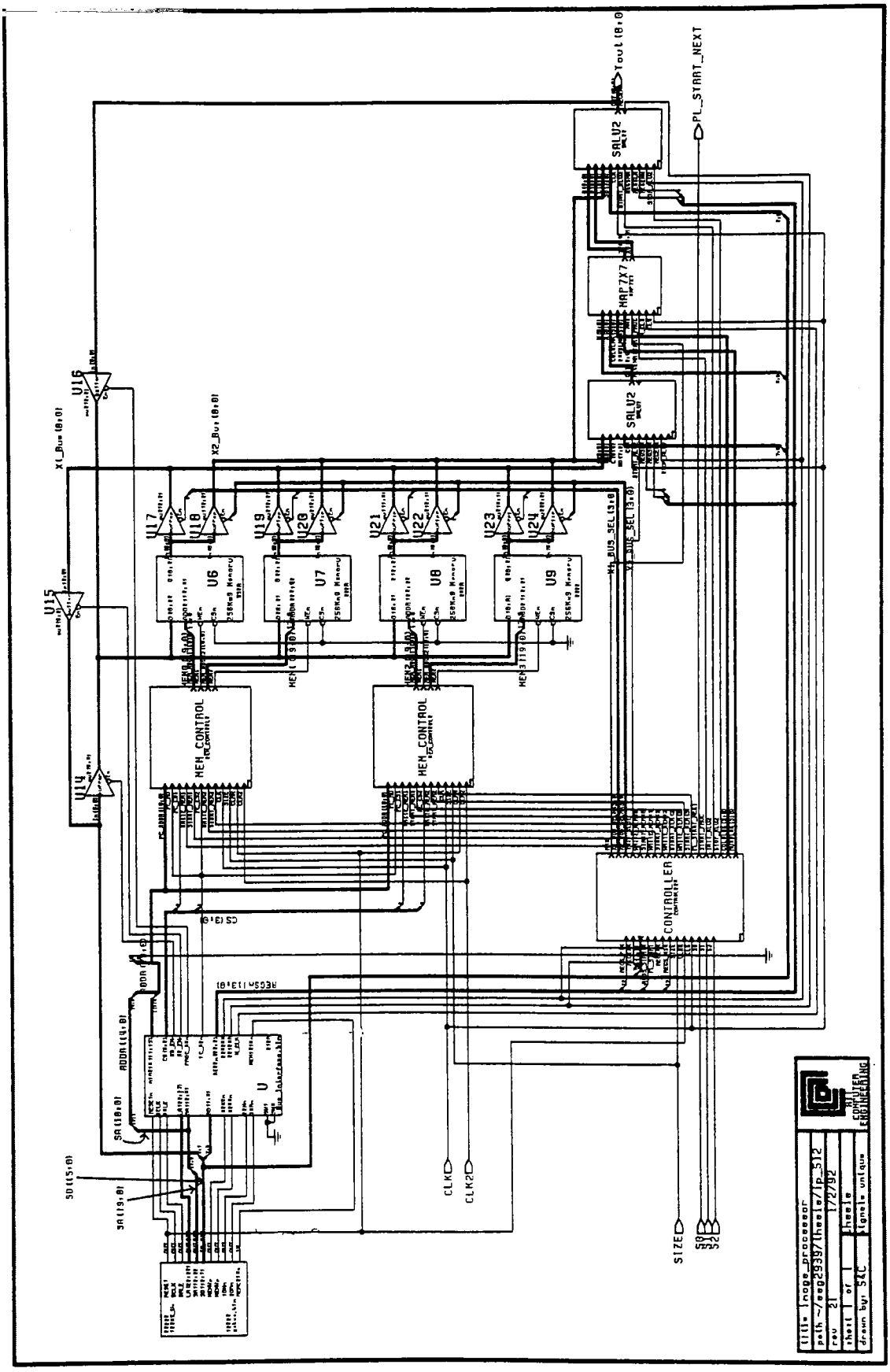

Figure 27 Image_Processor Schematic

Unlike Larry Rubin's single chip MAP layout[1], which was hard pressed to fit into the largest MOSIS die size of 7.9x9.2 mm, the partition of the MAP controller into two controls chips, the controller (Controller) and the memory controller (Mem_Control), was based on the pin limitations of the MOSIS standard chips not on silicon area constraints. Due to the large number of control and addressing lines involved, it was decided that the Controller and Mem_Control logic would stay partitioned in a manner similar to the ACTEL gate array design of Jens Rodenberg[3]. With 54 control lines and 8 power and ground lines the Controller would fit on a standard MOSIS 64 pin pad ring (see Figure 28 and Figure 29). A single memory controller with 49 address and control lines and 8 power and ground lines, could have also been fit into the standard 64 pin ring (see Figure 37). But, because the addition of another memory controller onto the chip only necessitates adding another 24 lines, due to both memory controllers using the same incoming 20 address lines, it was decided to use a standard MOSIS 84 pin pad ring to allow two memory controllers to be included on one chip (see Figure 35 and Figure 36). The 84 pin pad ring and 64 pin pad ring are both standard configurations for the MOSIS 4.6x6.8 mm intermediate die size.

Inputs	Description
REGS_PIn	Processor instruction and status register (active low)
REGS_MSn	Memory selection register (active low)
REGS_STARTn	Start register (active low)
REGENW	Enable write to registers
REGENR	Enable read from registers
SD(7:0)	Data bus for register reads and writes
CLK	Main Clock
CLRn	Global clear at system power-up (active low)
PL_START	Pipeline start input from previous stage
S(2:0)	Variable delay select
SIZE	Size selection bit (512, 1024)

Outputs	Description
START_MEM(3:0)	Start specified Mem_Control
WRITE_MEM(3:0)	Instruct specified Mem_Control of write operation
X1_BUS_SELn(3:0)	Assign specified memory to X1_Bus
X2_BUS_SELn(3:0)	Assign specified memory to X2_Bus
INIT_ALU1	Start ALU1
MAX	Specify desired morphological operation
START_PROC	Start MAP processing
ROWBLNK(5:0)	Instruct MAP to blank a particular row
COLBLNK(5:0)	Instruct MAP to blank a particular column
INIT_ALU2	Start ALU2
STOP_ALU2	Stop ALU2
PL_START_NEXT	Pipeline start output to next stage

Table 5 Controller Inputs / Outputs

Table 5 briefly describes the input and output control signals for the Controller chip. Each signal will now be described in greater detail.

- *REGS_PIn* Selects the processor instruction and status register which contains two write only bits for the MAP and bus mode instructions, and two read only bits for the process status bits. Bit 0, the MAP instruction bit, holds the MAX signal to the Controller. Bit 1 is the bus mode selection which determines whether the X2_Bus will be the input of ALU1 (mode = 0) or ALU2 (mode = 1). Bit 6 is the process status bit which is set by the Controller when the processor is ready to accept the next instruction. Bit 7 is the status bit which is set by the Controller when the processor is done processing and is ready to start again.

- *REGS_MSn* Selects the memory select register that is used to inform the Controller which memory or memories will contain the source image(s), which memory the output image will be written to, and which memory or memories will not be used. The Controller uses this information to route memory control signals to the appropriate Mem_Control chip, to select the bus buffers for processing operations, and to allow the host computer to read from on-board memory.

- *REGS_STARTn* Selects the start register to start the processor when the *REGENW* signal is also active. The contents of the data bus are insignificant when issuing the *REGS_STARTn* signal.

- *REGENW* Enables the host computer to write to the selected register. The Controller has three writable registers: the processor instruction and status register, the memory select register, and the start register. These are selected by active REGS_PIn, REGS_MSn, and REGS_STARTn signals respectively.

- *REGENR* Enables the host computer to read from the selected register. The Controller has one readable register, the process instruction/status register, which is selected by an active *REGS_PIn* signal.

- *SD(7:0)* The 8-bit data bus for register reads and writes.

- *CLK* The main system clock.

- *CLRn* Clears all flip-flops upon system power-up.

- *PL_START* Pipelined input from the previous processor stage to start the processor. This input effects the same result as the REGS_STARTn and REGENW signals asserted active.

- *S(2:0)* These bits set the variable delay length within the Controller from zero to seven clock cycles, that in addition to the fixed seven clock cycle delay, allow the Controller to work with the single chip MAP, seven chip MAP, or the ACTEL board MAP designs.

- *SIZE* This bit is set to zero for 512x512 pixel images and one for 1024x1024 pixel image processing.

- *START_MEM(3:0)* Issues a start signal to the Mem_Control chip(s) specified by the active bits. This will instruct the selected Mem_Control to start its memory address counter.

- *WRITE_MEM(3:0)* Instructs the Mem_Control chip specified by the active bit that its associated memory will be written to, starting when the *START_MEM* signal is issued. This will cause the selected Mem_Control to issue write enable signals to its memory chip during valid memory addresses.

- *X1_BUS_SELn(3:0)* Assigns the memory specified by the active bit to the X1_Bus by enabling the buffer connected between the memory's output and the bus.

- *X2_BUS_SELn(3:0)* Assigns the memory specified by the active bit to the X2_Bus by enabling the buffer connected between the memory's output and the bus.

- *INIT_ALU1* Instructs ALU1 to load its next instruction upon the next rising clock edge. This initializes ALU1 at the same time that the first valid image pixel is in its operand.

- *MAX* Informs the MAP that the next operation is an erosion (MAX = 0) or a dilation (MAX = 1).

- *START_PROC* Informs the MAP that the first valid image pixel will be in the target position upon the next rising clock edge. The MAP uses this signal to latch the windowing values and the desired morphological operation for the next image processing operation.

- *ROWBLNK(5:0)* Informs the MAP which rows are to be blanked.

- *COLBLNK(5:0)* Informs the MAP which columns are to be blanked.

- *INIT_ALU2* Instructs ALU2 to load its next instruction upon the next rising clock edge. This initializes ALU2 at the same time that the first valid image pixel is its operand.

- *STOP_ALU2* Informs ALU2 that its operand upon the next rising clock edge will no longer be a valid pixel for the current processing operation.

- *PL_START_NEXT* Pipeline start output to the next image processor.

Inputs	Description
PC_ADDR(19:0)	20-bit address from host computer
PC_RD	Read enable from host computer
PC_CS1	Mem_Control 1 Chip selection from host computer
START_MEM1	Mem_Control1 Local memory address counter start signal
WRITE_MEM1	Mem_Control1 Write enable for image processing operations
PC_CS2	Mem_Control2 Chip selection from host computer
START_MEM2	Mem_Control2 Local memory address counter start signal
WRITE_MEM2	Mem_Control2 Write enable for image processing operations
CLK	Main clock
CLK2	Second clock for proper write enable timing
CLRn	Global clear at system power-up (active low)
SIZE	Size selection bit (512, 1024)

Outputs	Description
MEM_ADDR1(19:0)	20-bit image memory address for Mem_Control 1
WEn1	Write enable for image memory control 1 (active low)
MEM_ADDR2(19:0)	20-bit image memory address for Mem_Control 2
WEn2	Write enable for image memory control 2 (active low)

Table 6 Mem_Control Inputs / Outputs

Table 6 briefly describes the input and output control signals for the Mem_Control chip with two memory controllers. Each signal will now be described in greater detail. For those signals duplicated for memory control 1 and memory control 2 on the chip, only one explanation is provided.

- *PC_ADDR(19:0)* The 20-bit address used by the host computer to access on-board memory.

- *PC_CS* The signal from the host computer to allow the host computer or Mem_Control chip to access memory.

- *PC_RD* The signal from the host computer to specify that the host computer will perform a memory read or memory write.

- *START_MEM* Issued by the Controller to control the internal address counter of each Mem_Control.

- *WRITE_MEM* Issued by the Controller to control the internal address counter of each Mem_Control.

- *CLK* The main board clock.

- *CLK2* The main CLK signal delayed by approximately 30 nsec. to enable proper timing of the WEn (write enable) signal to the memory.

- *CLRn* The system power-up signal to clear the Mem_Control chip.

- *SIZE* This bit is set to zero for 512x512 pixel images and one for 1024x1024 pixel image processing.

- *MEM_ADDR(19:0)* The address used for memory access.

- *WEn* The write enable signal to specify that a memory read or write will take place.

3. The Logic Design of the Controller and Mem_Control

The Controller logical design consists of the bus interface registers, the X1 and X2 bus selection logic (Bus_Select), the memory control selection logic (Mem_Select), the counter to start blanking (Start_Blank), the blanking counter (Blank_Counter), and the delay logic to generate the ALU and MAP control signals (Var_Delay). The bus interface logic consists of three registers, the processor instruction and status register, the start register, and the memory select register. The processor instruction and status register contains two read only bits set by the Controller when *INIT_ALU2* or *STOP_ALU2* are active. When read by the host using an active *REGENR* signal, these bits alert the host computer that the processor is ready to accept the next instruction or processing has completed. The processor instruction and status register also contains two write only bits that provide the *MAX* signal from the host computer to select which processor operation is to be performed (erosion or dilation), and the X2_Bus mode bit to select whether the X2_Bus will be the input to ALU1 or ALU2. The start register is a single bit write only register, which when paired with the *REGENW* signal active, starts the processor with a *START* signal. When the *START* signal is generated the Start_Blank logic block begins counting. The Start_Blank counters count is set to match when the first pixel of the image is about to be acted upon by the MAP. When the final count is reached, the Start_Blank *Done* signal is generated. This causes the Blank_Counter to begin counting and generating *ROWBLNK* and *COLBLNK* signals for the MAP, this signal is also passed on to the MAP as the *START_PROC* signal. When the Blank_Counter final count is reached, the Blank_Counter *Done* signal is generated. Additional timing signals used by the ALUs and MAP, such as *INIT_ALU2*, *STOP_ALU2*, *START_Y*, and *PL_START_NEXT*, are generated by the variable delay logic from the

Start_Blank and Blank_Counter *Done* pulses. The memory select register is an 8-bit write only register used by the host computer to control the Bus_Select and Mem_Select logic. Control signals are passed in from the host computer, and decoded with the bus and memory select logic, to determine which memories will take part in the operation and the X2_Bus operation mode. Refer to Figure 28 through Figure 34 for the Controller design schematics.

The Mem_Control logical design consists of the memory address counter (Mem_Counter), the write enable (*WEn*) logic, and a 20 wide 2-input MUX. Mem_Counter is synchronous counter that is reset by either an active *START_MEM* or *CLRn*. Its function is to step through every memory address for internally generated memory reads or writes. The *WEn* decode logic enables memory reads or writes from the host computer, when *PC_CS* is active *WEn* simply equals *PC_RD* supplied from the host computer delayed by *CLK2*. When *PC_CS* is not active then *WEn* is started to toggle by an active *START_MEM* and *WRITE_MEM*, it continues to toggle until the Mem_Counter *Done* pulse is received. Once again it is delayed by *CLK2* from the main *CLK* to allow proper timing for the memory. *PC_CS* is also to used by the 20 wide 2-input MUX to select the addressing from the *PC_ADDR(19:0)* or the output of Mem_Counter *ADDR(19:0)*. Refer to Figure 35 through Figure 39 for the Mem_Control design schematics.

The logic designs of the Controller and Mem_Control retained much of the architecture of the Actel gate array design of Jens Rodenberg, while adding the ability to selectively process 512x512 or 1024x1024 pixel images using a "size" bit to perform the selection. Multiplexer decode circuits were added to Mem_Control, Start_Blank, and Blank_Counter using the "size" bit to selectively change *Done* pulse timing for 512x512 versus 1024x1024 pixel images (see Figure 33, Figure 34, and Figure 37). For the CMOS chip version, two Mem_Control circuits were combined onto one chip to reduce the chip count.

The three counter circuits Start_Blank, Blank_Counter, and Mem_Counter were redesigned using synchronous binary counters (see Figure 33, Figure 34, and Figure 38). This design is simpler than the Actel design and takes advantage of the faster ripple delay path through the custom CMOS cells than the Actel gates. The ripple carry path still settles out in less than half a clock cycle. The row and column blanking decode logic within the Blank_Counter was also simplified and modified to take advantage of the faster CMOS cells to perform a combinational logic decode of the counter lines, rather than using shift registers propagating a pattern, to generate blanking pulses during the same clock cycle in which they are used in the MAP. This removed one register delay cycle in the Controller design and was possible because the blanking signals are not used until the end of the clock cycle within the MAP. The *WEn* decode logic in Mem_Control (see Figure 37) was simplified from the Actel designs pair of 4-input MUXes and DFF to a NOR and 2-input MUX. Refer to Table 7 for the truth table used to simplify the logic. The simplified logic equation is (where an underscore refers to negation):

$$WEn = \underline{START\_MEM}*\underline{Done}*WEn + START\_MEM*WRITE\_MEM$$

	START_MEM /Done			
WEn/WRITE_MEM	**00**	**01**	**11**	**10**
00	0	0	0	0
01	0	0	1	1
11	1	0	1	1
10	1	0	0	0

Table 7 Write_Enable Truth Table

The bus selection and *WRITE_MEM*, *START_MEM* memory selection decodes in the Controller (see Figure 31 and Figure 32) were simplified due to the better part selections in the CMOS cell library than the Actel parts library. The miscellaneous Controller logic was simplified mainly by using the RIT Standard Cell Library components to replace the Actel library components. The *MAX* and *START_PROC* signal outputs from the Controller consisted of three identical outputs in the Actel board design to provide sufficient fanout for the 23 MAP chips of that design. With the single chip VLSI MAP both of these signals have been reduced back to a single output.

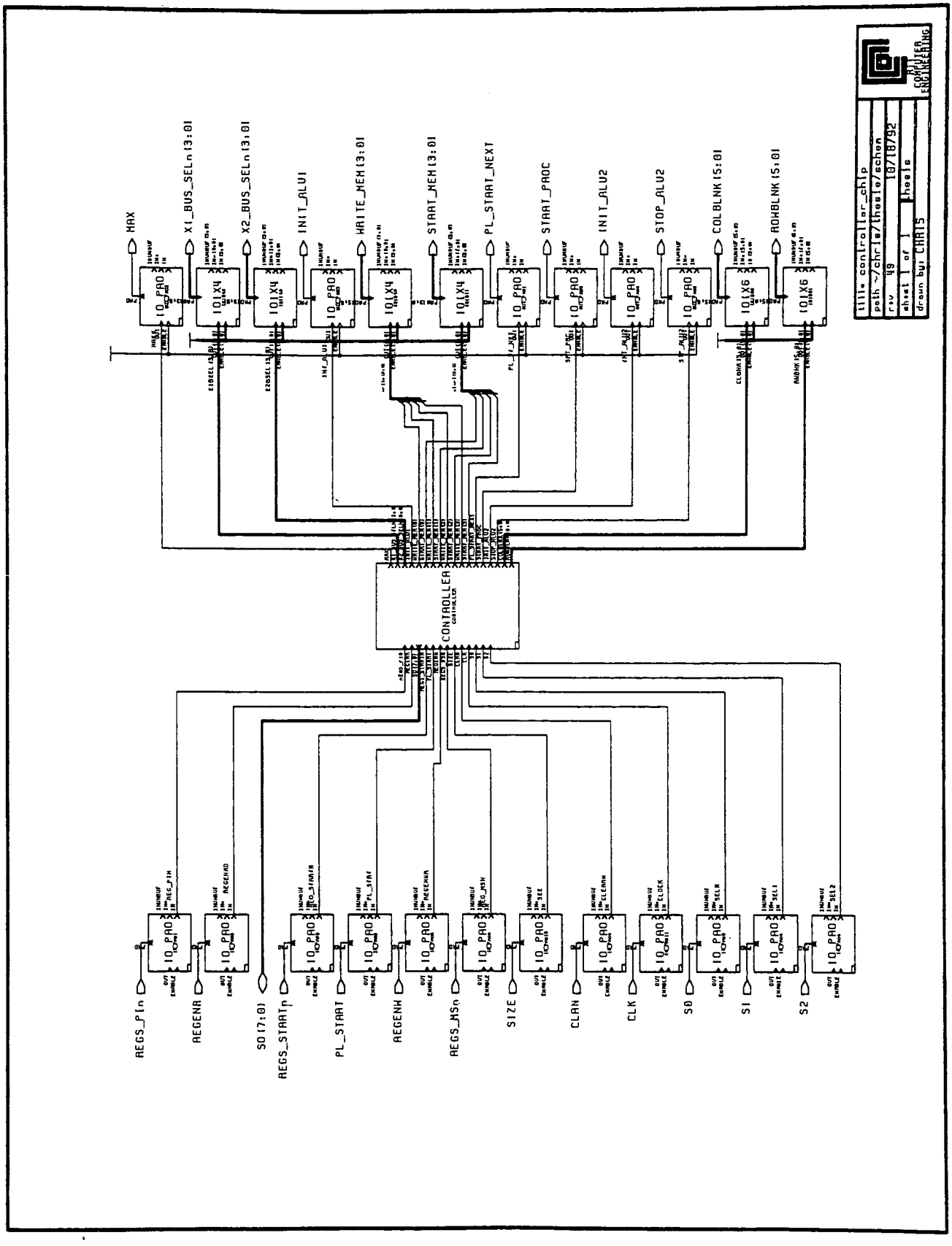

Figure 28 Controller_Chip Schematic

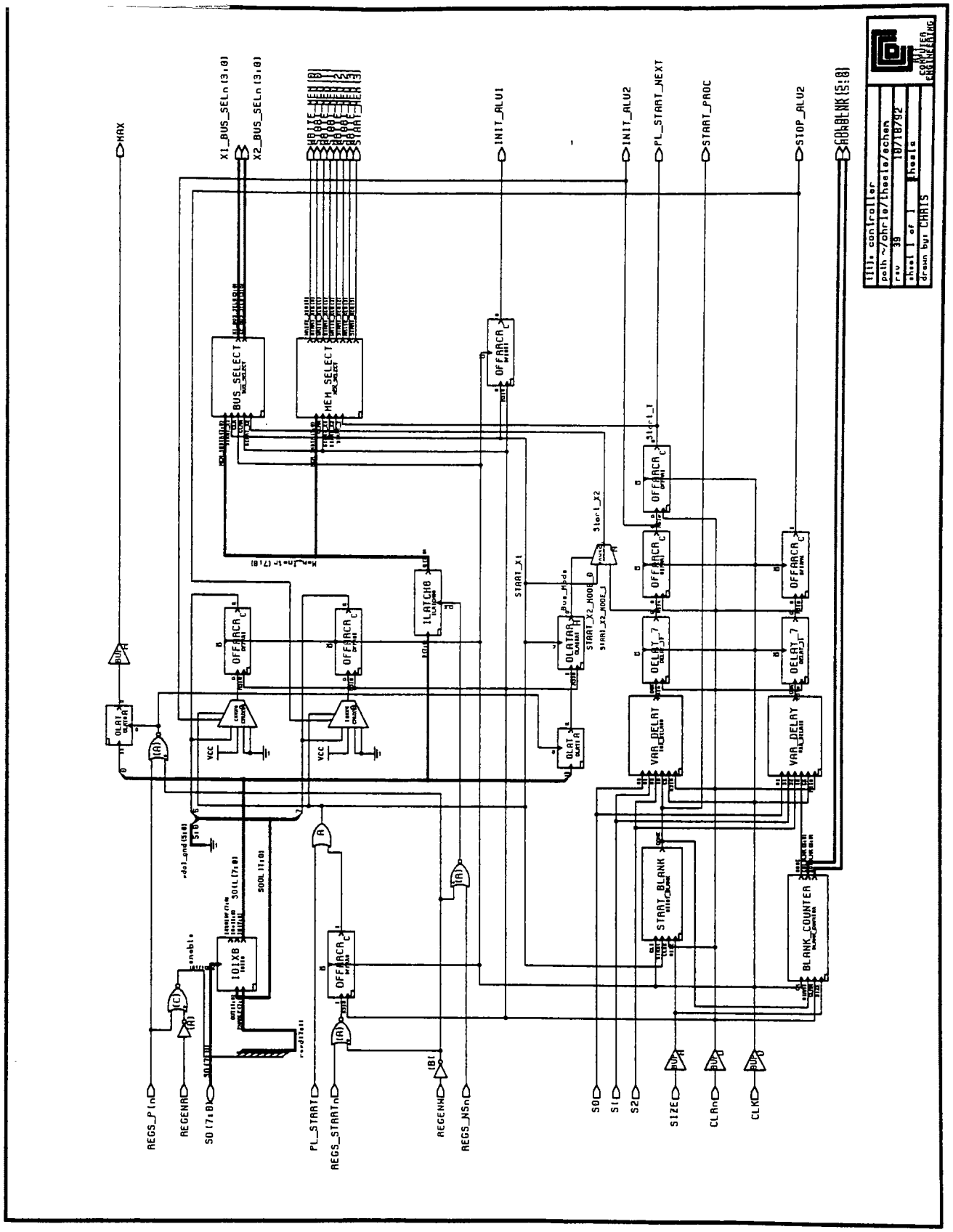

Figure 29 Controller Schematic

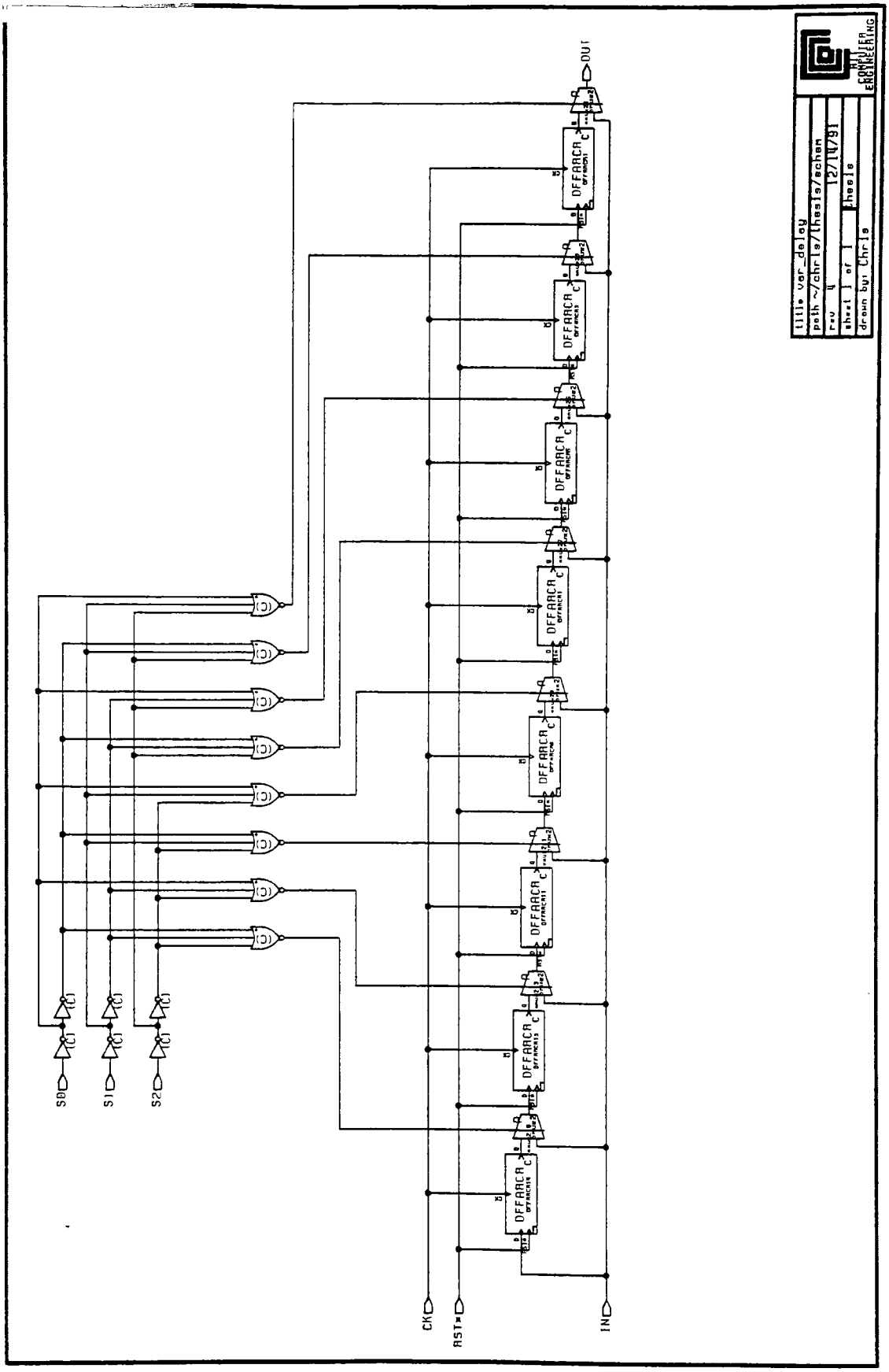

Figure 30 Var_Delay Schematic

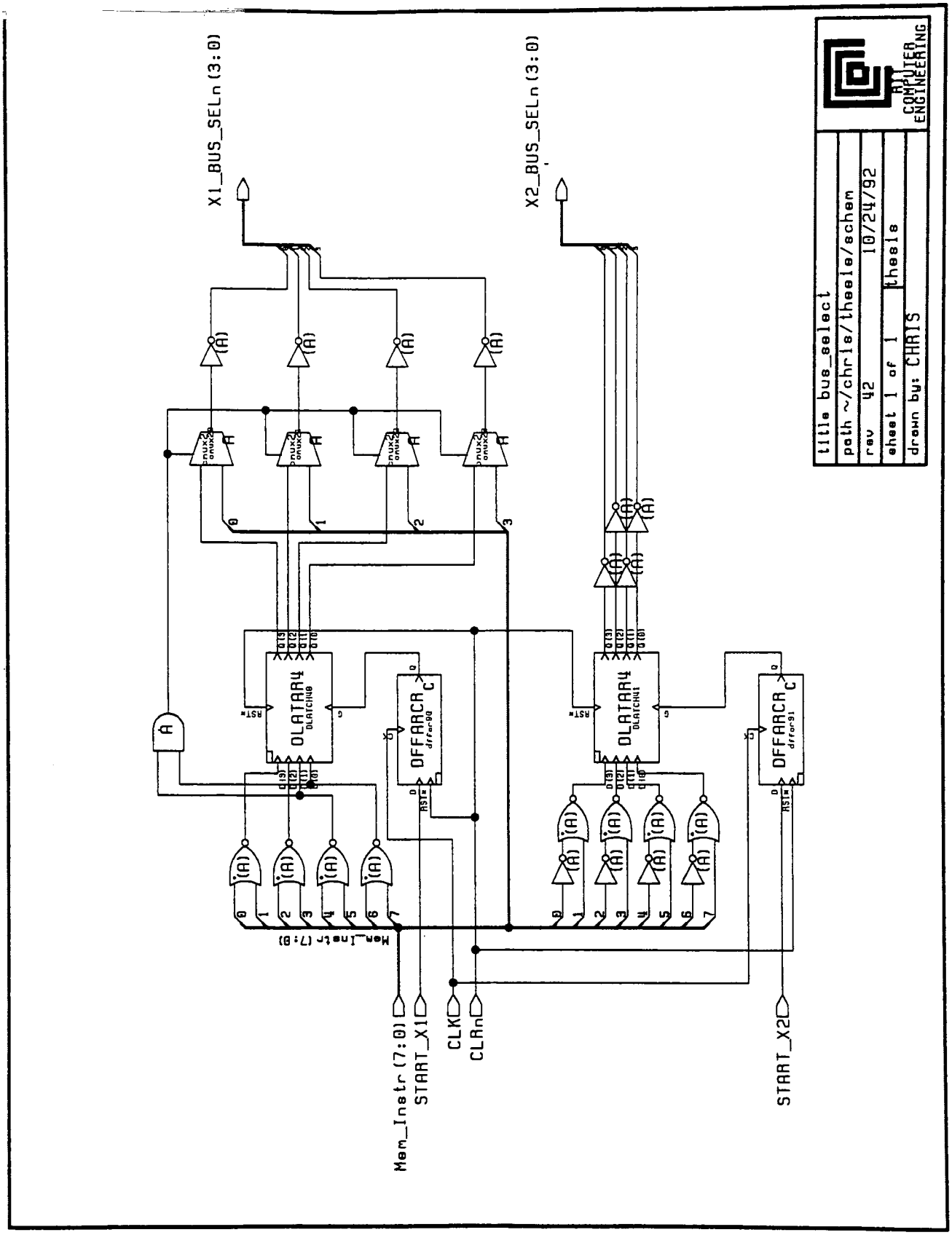

Figure 31 Bus_Select Schematic

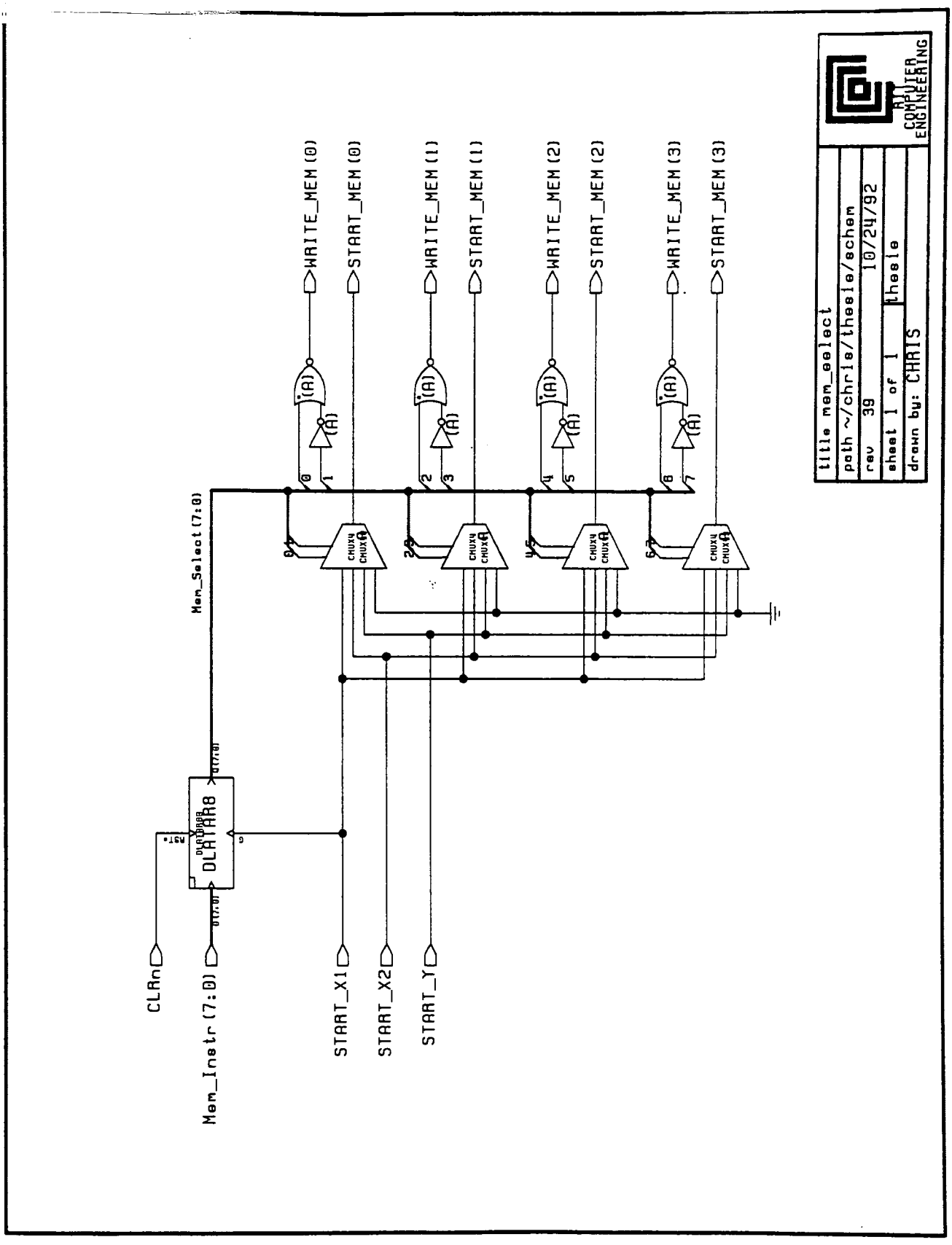

Figure 32 Mem_Select Schematic

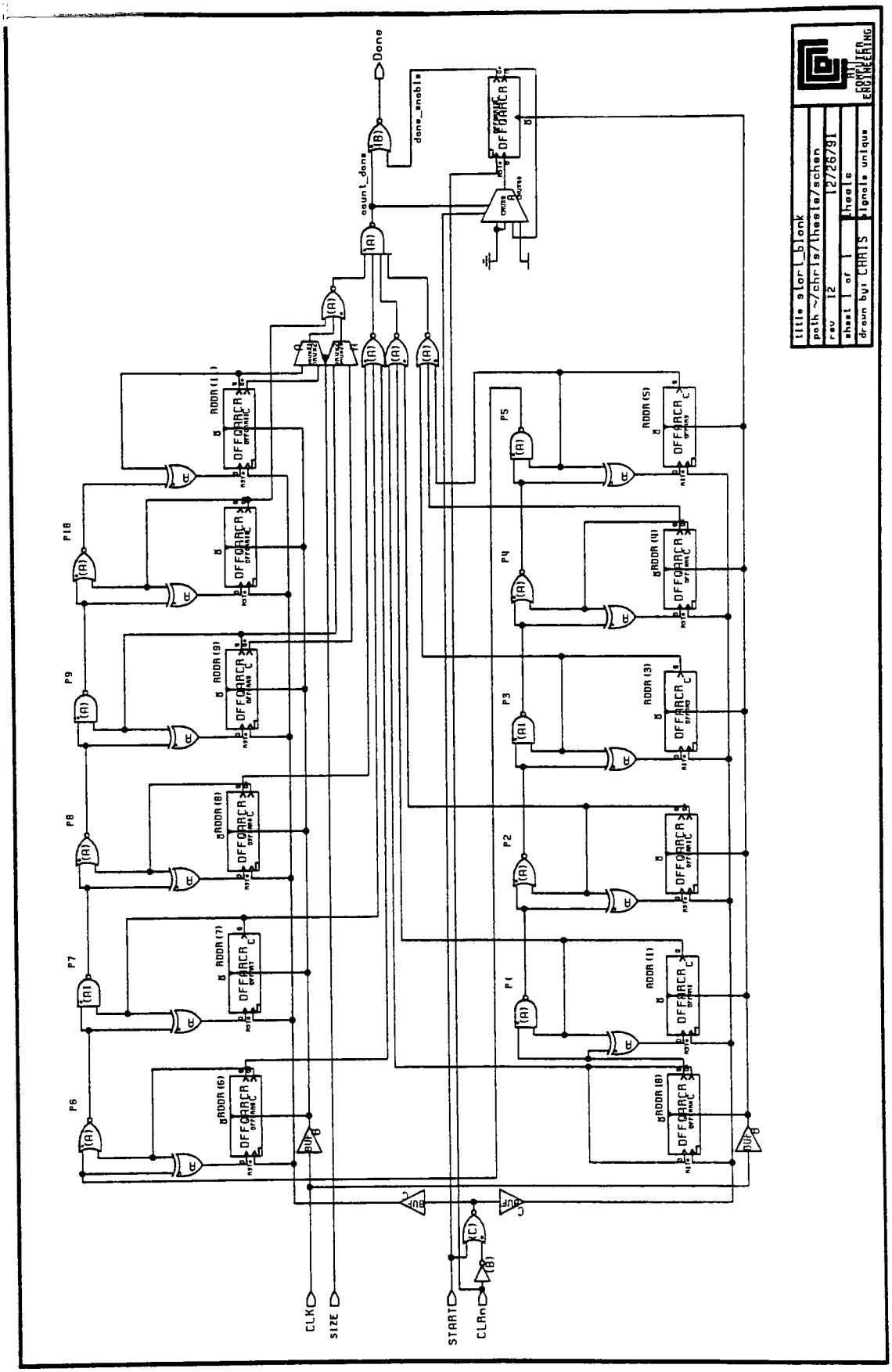

Figure 33 Start_Blank Schematic

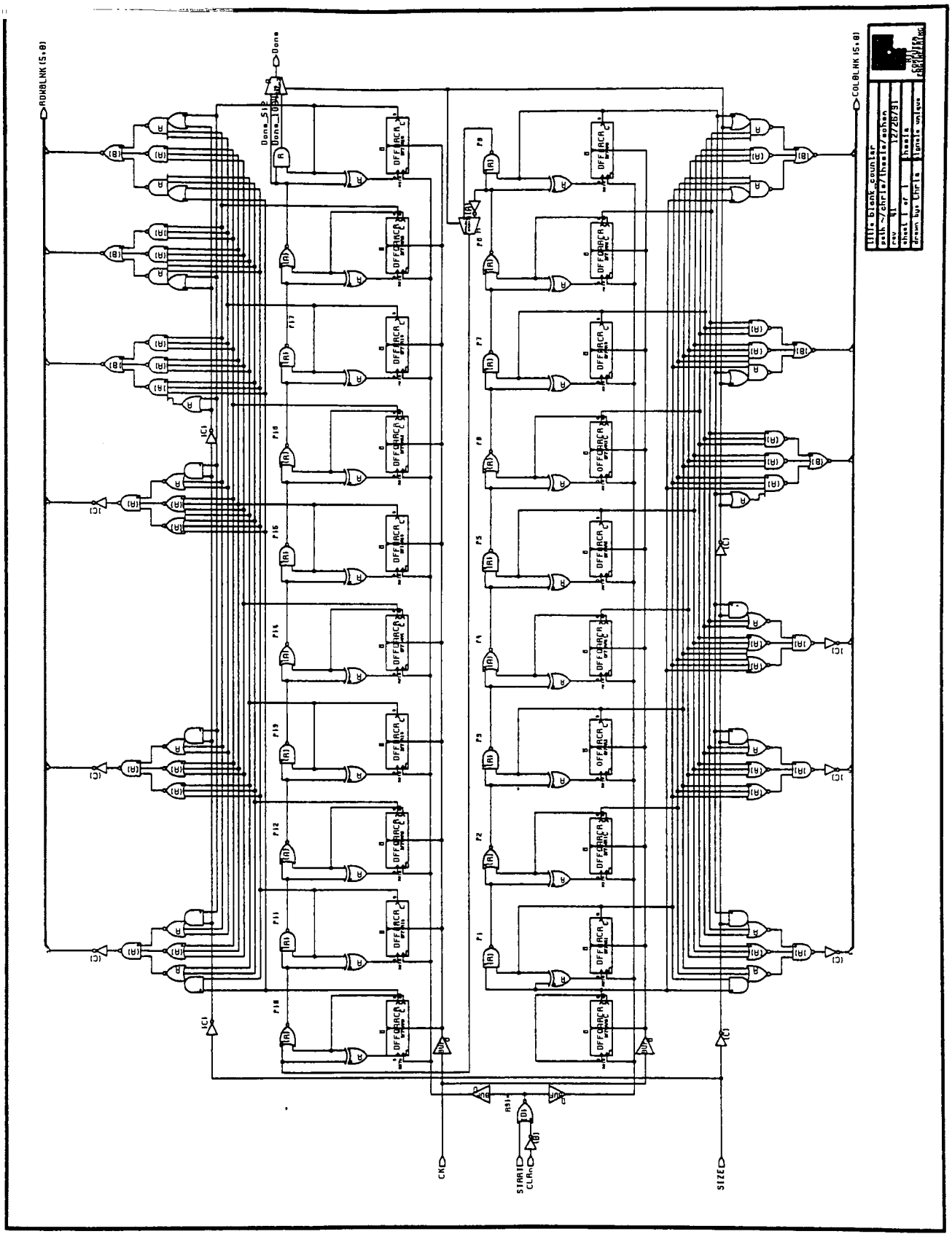

Figure 34 Blank_Counter Schematic

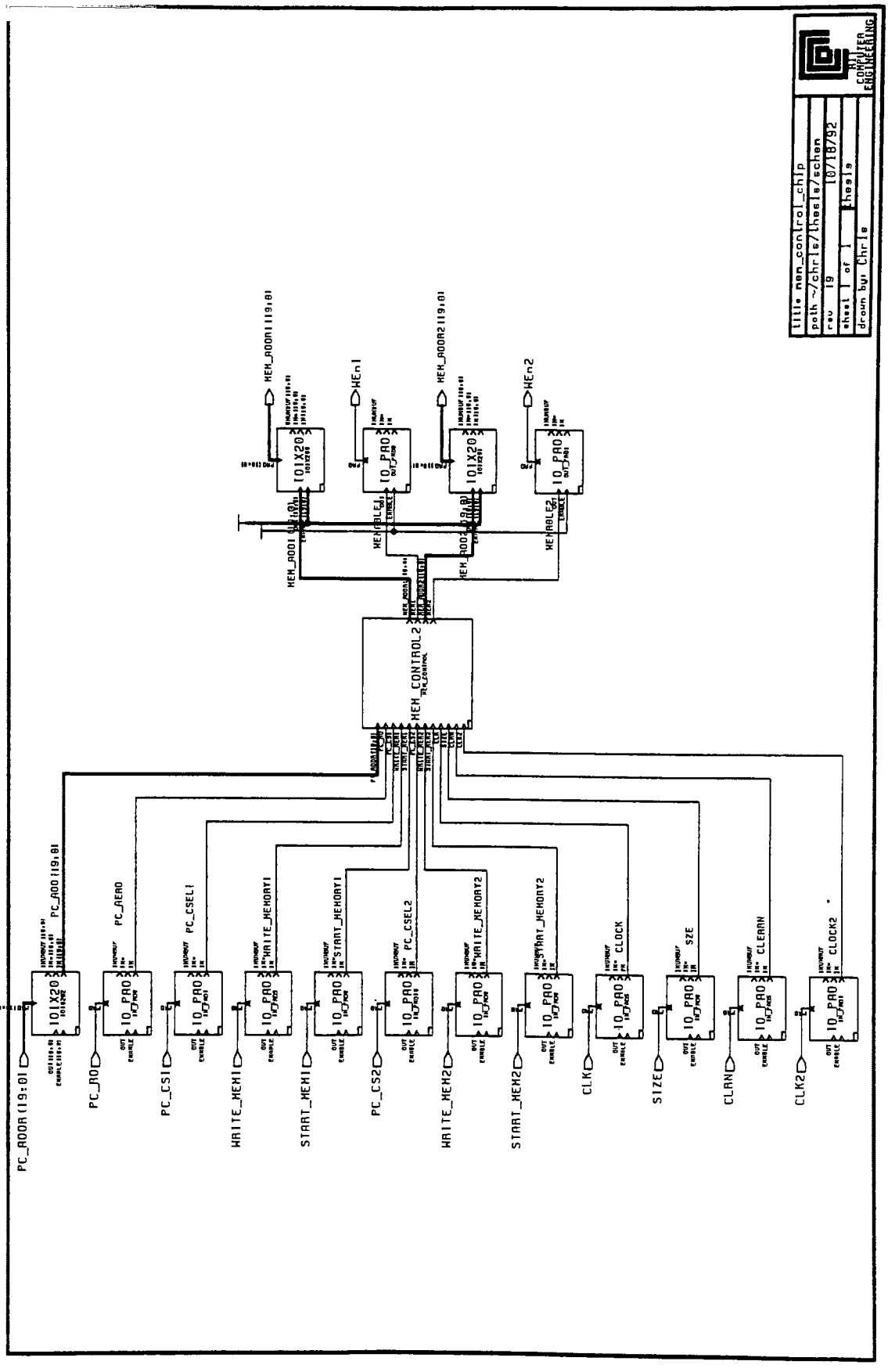

Figure 35 Mem_Control_Chip Schematic

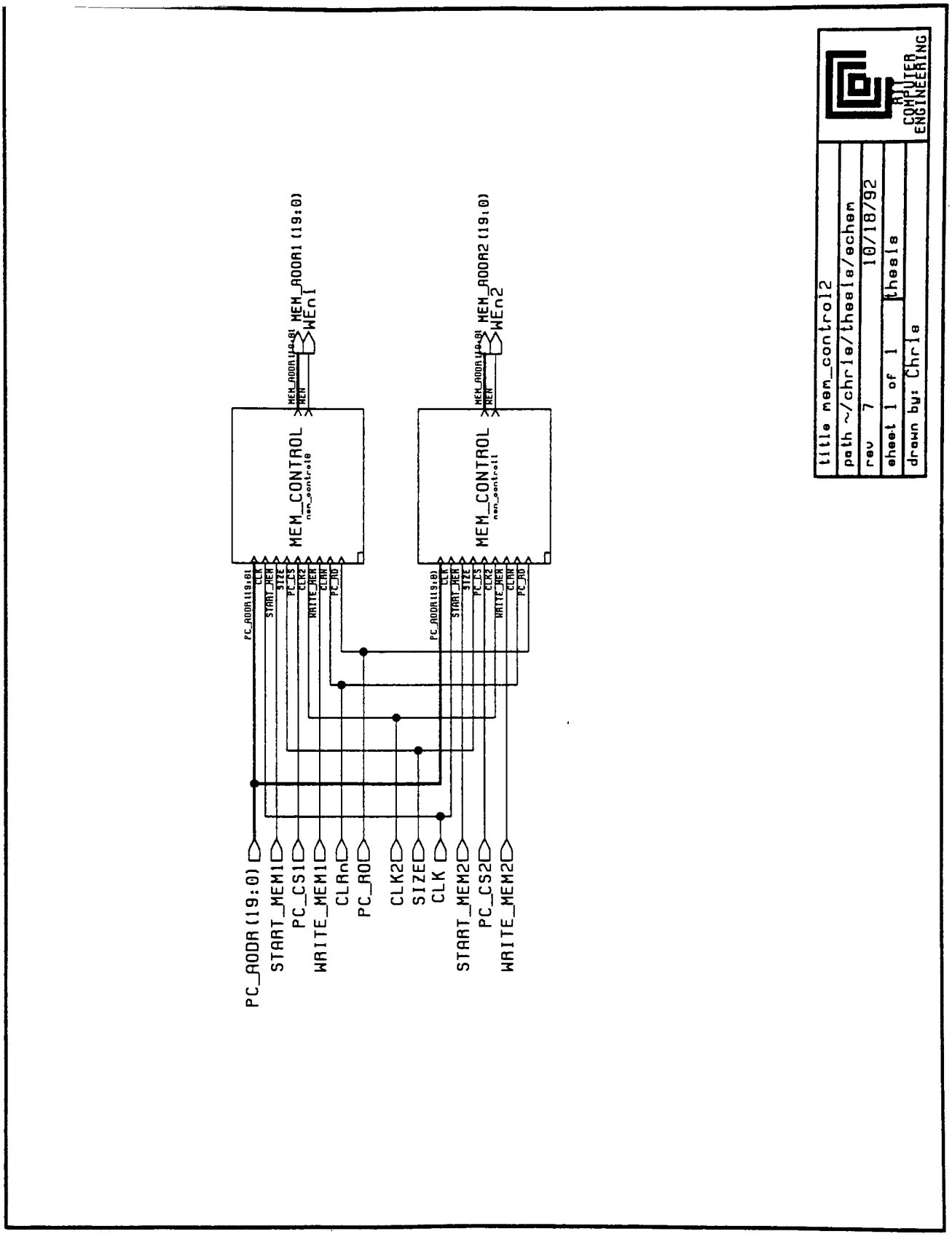

Figure 36 Mem_Control2 Schematic

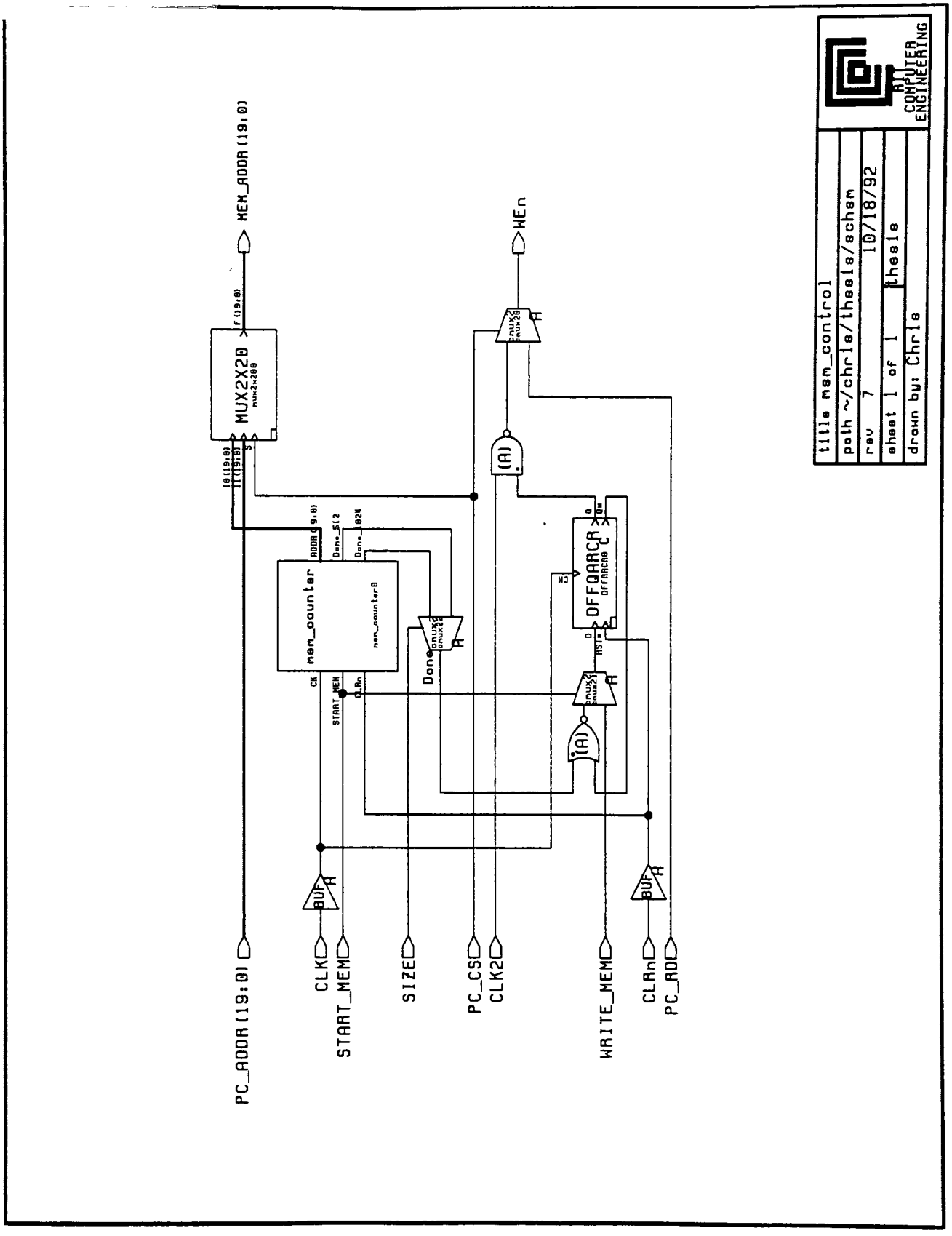

Figure 37 Mem_Control Schematic

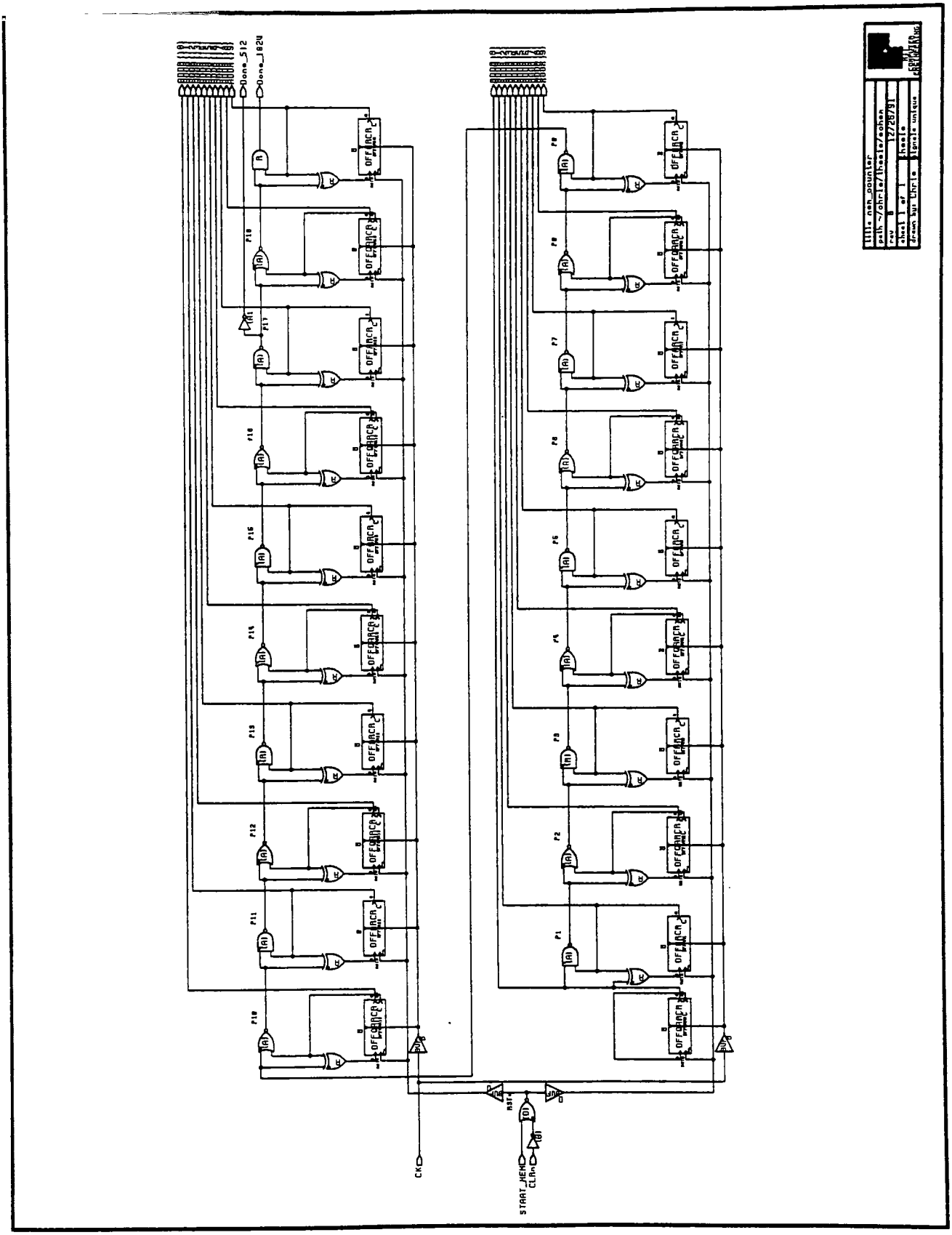

Figure 38 Mem_Counter Schematic

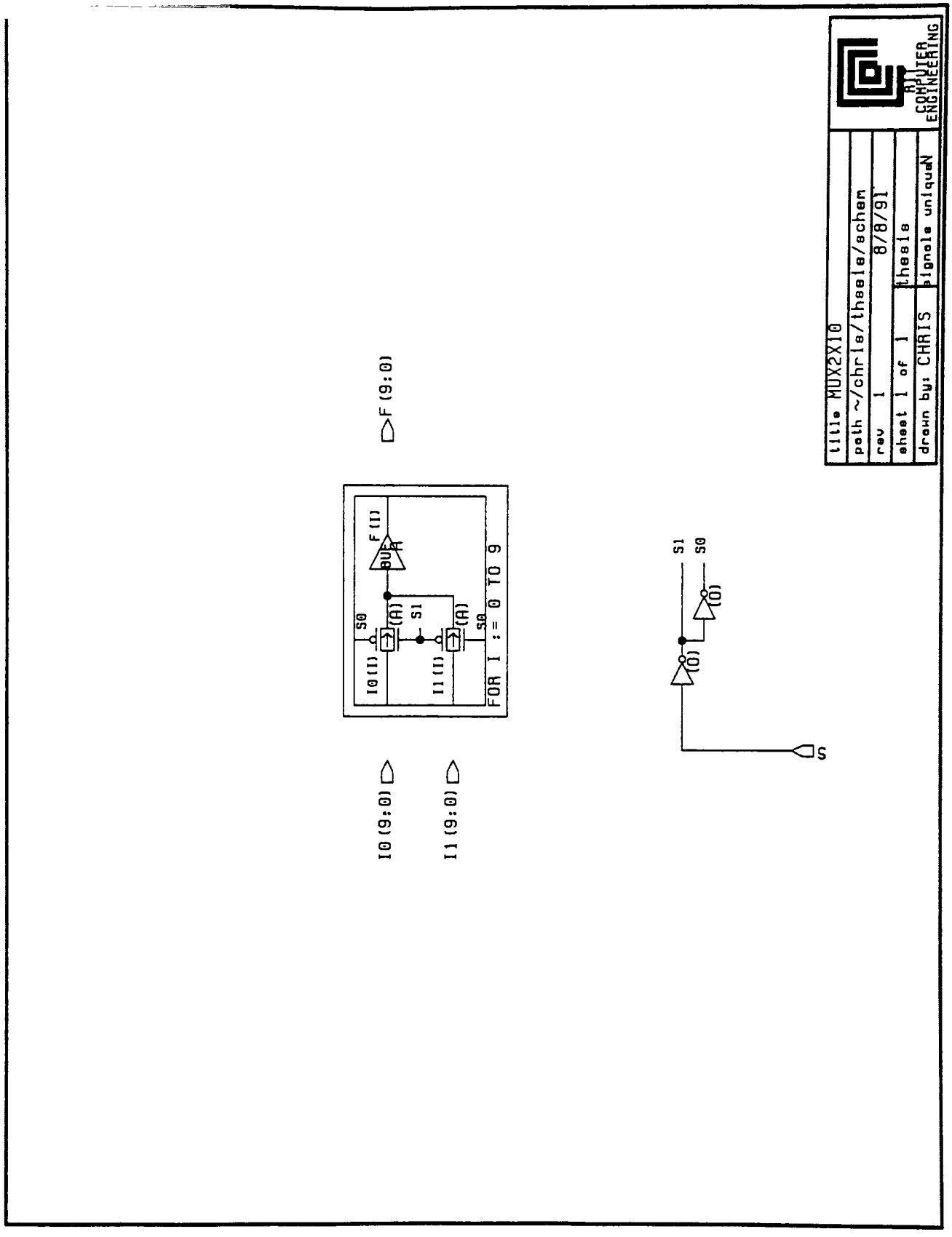

Figure 39 MUX2x10 Schematic

4. The Logic Simulation of the Controller and Mem_Control

Logical simulation consisted of testcases to verify the correctness of the redesigned circuitry with Jens' original design as well as the modified register delays due to less latency in the CMOS MAP logic. The three counter circuits Start_Blank, Blank_Counter, and Mem_Counter were simulated first to verify that the *Done* pulses for 512 versus 1024 pixel images were generated at the right number of clock cycles and were single clock width pulses. For the Start_Blank *Done* pulse this was 512x3 rows + 4 + 2 ALU - 1 = 1541 clock cycles later for a 512 pixel image. Refer to Appendix D, 3. Start_Blank Logical Simulation File and 4. Start_Blank Logical Simulation Results Figure 43 for the Accusim simulation data. Then the Blank_Counter's *ROWBLNK* and *COLBLNK* blanking decode logic was simulated to verify the new decode logic was correct. The top level Mem_Control logic was simulated next for one 256Kx9 memory block to verify the address lines changed on the correct clock cycles and that *WEn* toggled once for each clock cycle from *START_MEM* to Mem_Counter *Done* (512x512 cycles). Figure 40 shows the general timing of the Controller and Mem_Control depicting the sequence from *WRITE_MEM* to Mem_Counter *Done*. *WRITE_MEM* is asserted prior to *START*, *START_MEM* is asserted 1557 cycles after *START*, *WEn* begins to toggle once *START_MEM* falls until Mem_Counter *Done* falls 512x512 cycles later. The top level Controller was simulated next with different testcases to verify the four memories could be selected correctly as well as the *X1* and *X2* buses. The Bus_Select logic was simulated to verify that *X1_BUS_SELn(3:0)* and *X1_BUS_SELn(3:0)* defaulted to a high output state when *CLRn* was active. The *INIT_ALU1*, *INIT_ALU2*, *START_PROC*, and *STOP_ALU2* signals were also verified to be generated after the correct number of latency cycles.

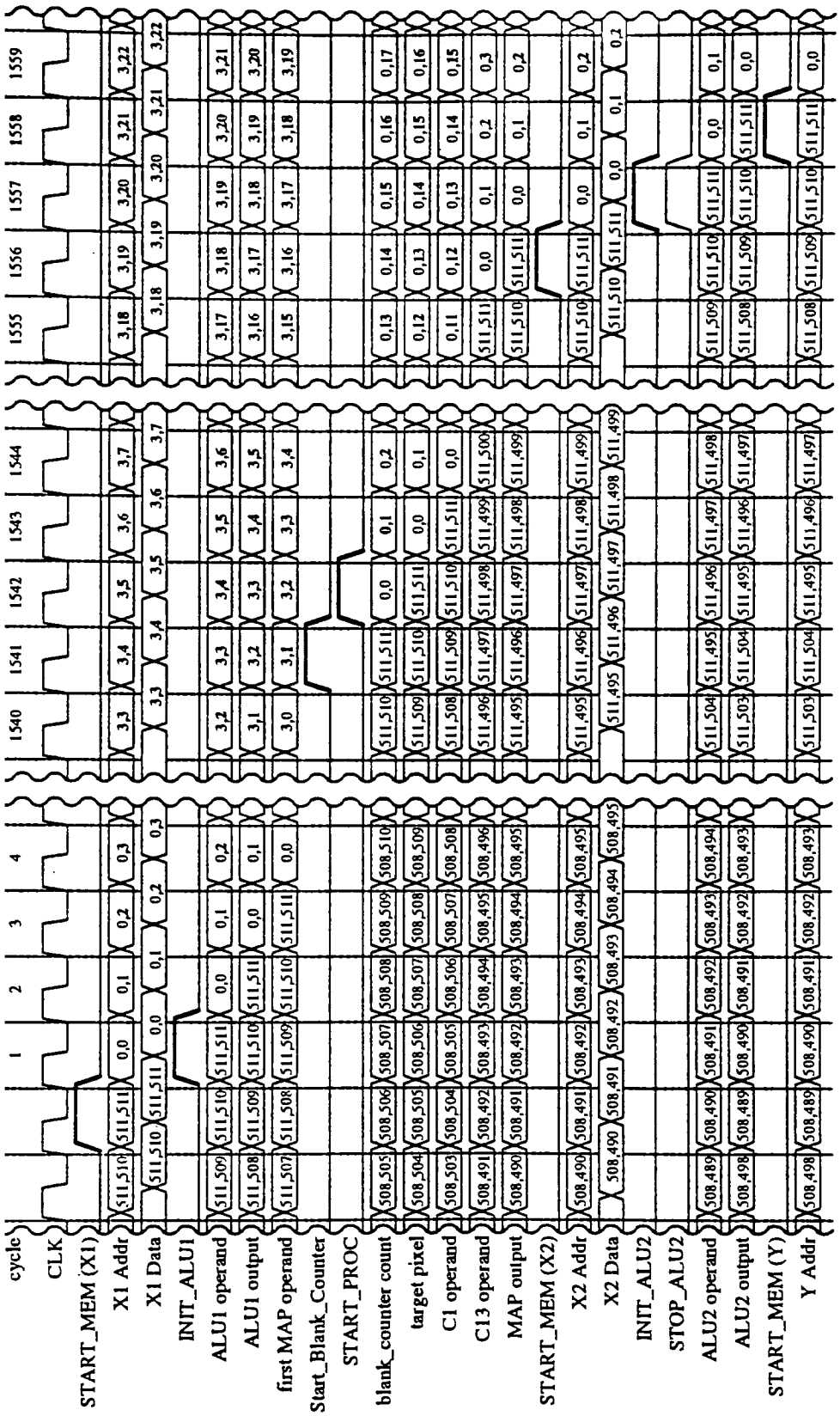

Figure 40 Control Timing Diagram

5. The Circuit Design of the Controller and Mem_Control

The circuits for the Controller and Mem_Control were designed to take advantage of the RIT Standard Cell Library parts available. During the circuit design stage it was necessary to design additional components and standard cells to allow the design to be optimized and to minimize critical path delay times. The three counter circuits Start_Blank, Blank_Counter, and Mem_Counter were originally designed using D flip-flops, XOR, and AND gates to form synchronous binary counters. OR-AND-Invert (see Figure 8 ND3_2 Schematic) and AND-OR-Invert (see Figure 9 NR3_2 Schematic) cells were designed for the Blank_Counter's *ROWBLNK* and *COLBLNK* blanking decode circuitry to keep all decode path gate delays to three or less (see Figure 34). Additional D Latches, D flip-flop and transmission gate MUXes were also designed (see Figure 5 through Figure 7) to allow the design to be optimized as well as sized for drive strengths.

6. The Circuit Simulation of the Controller and Mem_Control

Circuit simulation was done with critical paths in mind to verify the design would work at a 16 MHz clock frequency. All simulations were performed with a 20 MHz clock and Mil Spec conditions, i.e. Vcc = 4.3 volts, temp = 125° C, 5 nsec. rise and fall times, and with Spice level 2 parameters. The longest delay paths were through the ripple carry paths of the counters. During the simulation phase the three synchronous counters Start_Blank, Blank_Counter, and Mem_Counter were redesigned to use D flip-flops with Q and Q bar outputs, XOR, and alternating NAND, NOR gates in the ripple path. This removed one gate delay per D flip-flop and cut the gate delay path in half from the original D flip-flop, XOR, AND gate design, which allowed

the counters to work at 20 MHz with a delay of 12.1 nsec. The delay was measured from the 50% rise time of the rising edge of the clock to the 50% rise time for the *Done* pulse. Refer to Appendix D, 1. Start_Blank Physical Simulation File and Figure 42 Start_Blank Logical Simulation Output for the Accusim simulation input file and results. The Start_Blank physical simulation results in Figure 42, and logical simulation results in Figure 43, together verify the clock cycle on which the *Done* pulse is generated and the delay between the *Done* pulse generation and the clock. The Blank_Counter's *ROWBLNK* and *COLBLNK* blanking decode logic delay paths were also simulated to verify the blanking pulses were settled before the second half of the clock cycle. Sizing was performed on all gate output driver stages to verify all rise and fall times were around 5 nsec. Spice simulation using Mentor's Accusim required initialization of all D flip-flops internal nodes between latches to known voltages to allow the initial DC analysis to converge. The internal nodes were set to 4.3V or 0V using the Accusim Initial Condition statement.

7. The Morphological Array Processor Top Level Simulation

After the Controller, Mem_Control and MAP were individually simulated logically and at the transistor circuit level, and debugged, the overall architecture of the MAP processor was simulated (see Figure 27). During this phase of simulation quite a few design errors were found and corrected. The *Done* pulse timing from Start_Blank was found to be off by one clock cycle due to an unaccounted for delay when the MAP uses the signal. During this phase of simulation, the CMOS MAP design underwent significant changes and it was decided to add a selectable variable register delay path to allow the Controller to compensate for different design's latency delays through the MAPs adder and comparator circuitry. The adder and comparator latency path compensation in the Controller had to be modified for the CMOS MAP

design. The CMOS design has a 7 gate delay versus the 14 gate delay in the Actel gate array design. A selectable delay from 7 to 14 clock cycles was implemented which allows use of the Controller with the single chip MAP design, the seven chip MAP design, and the Actel gate array MAP design (see Figure 30). The delay is determined by the pull-up or pull-down setup of the *S0-S2* MUX select bits. It was at this point that major logic errors were detected in the CMOS MAP design. The comparator circuitry for MAX and MIN were found to have been incorrectly designed to compare from LSB to MSB instead of vice versa and the design was found to have a problem with the last comparison bit incorrectly toggling the result in approximately 10 percent of the cases. The additional logic necessary for the MAP redesign prevented the MAP from being able to be layed out on a standard MOSIS die size. This forced a repartition of the MAP into seven identical chips each performing one row of additions and comparisons. A problem was also detected in the MAP blanking logic, when the result of an operation yielded a -512 result it was incorrectly treated as a negative infinity instead of -511, also some internal blanking lines were tied permanently to ground. When these design changes were incorporated by Shishir Ghate, the design top level was simulated using Jeff and Jens testcases, with the resultant data compared with Jens' test results data.

8. The Layout of the Controller and Mem_Control

All standard cell designs were simulated, and DRC and LVS checked, before being added to the RIT Standard Cell Library. This included the pad rings for the Tiny, 64, and 84 pin pad rings. The Controller and Mem_Control were then layed out using the RIT Standard Cells Library and the Cell Station tools, which were used for automatically placing and routing the designs. The resulting layouts easily fit on the 84 pin chip and since space on the chip was

not a problem, the packaging of the designs was more dependent on the number of pins required. Since space was not a problem, it was not necessary to use Cellsqueeze to compress the designs. Compressing the design can introduce a large number of errors that have to be corrected by the designer. Thus squeezing the design is not recommended unless space is a problem. The Cell Station layout needed to be cleaned up for DRC errors as well as unrouted and poorly routed lines, including looping, polysilicon over contacts, etc. Also long polysilicon lines needed to be replaced with metal lines and power grids required additional contacting cuts. The completed layout then had to be placed within the appropriate pad ring, layed out to MOSIS specifications, and hand routed for input and output signal connections. The power routing on both chips consisted of a tree routing to the logic strips, using two VCC and two VSS pads to supply internal power to the circuit. The outer power rings, supplying power to the input/output pad ring, also had two VCC and two VSS pads supplying them power, but the outer rings were electrically separate from the internal power rings to minimize electrical noise within the circuit. Figure 41 shows the power routing architecture within the controls chips. The layout again required DRC and LVS checking before a final mask could be made. To allow the ERC and LVS to complete successfully, it was necessary to put a single contact cut in to electrically connect the internal and pad ring VCC power lines and the internal and pad ring VSS power lines together so the rules checkers would see one VCC and one VSS node. Once the rule checking was completed these cuts were removed.

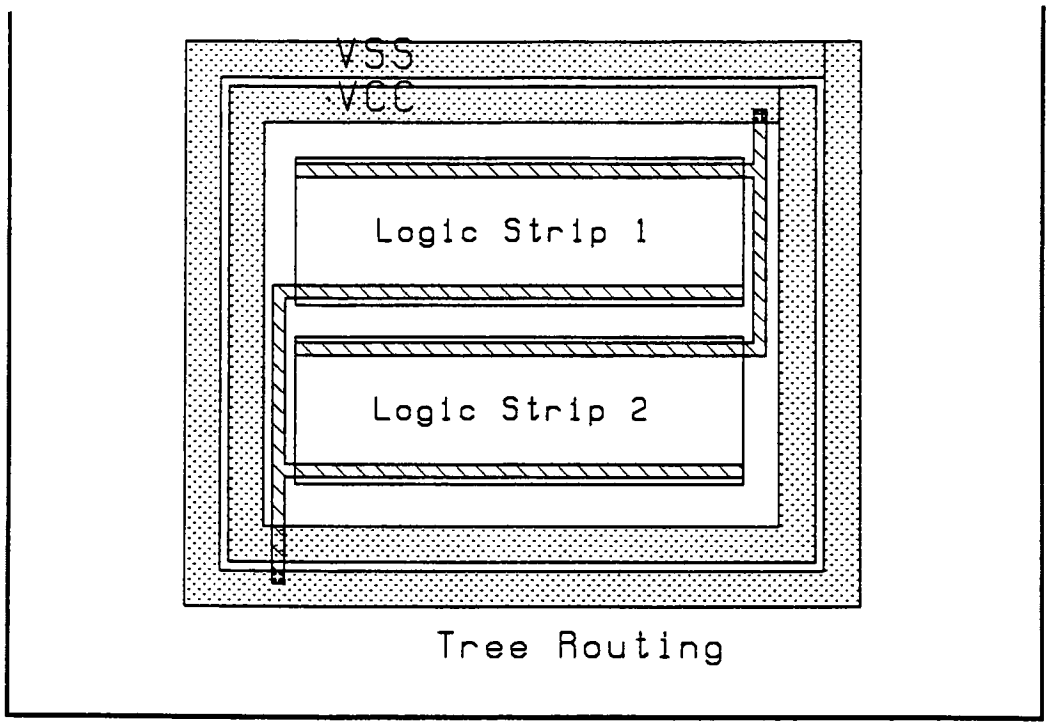

Figure 41 Power Routing Architecture

9. Suggested Testing Methodology for the Controller and Mem_Control

After the Controller and Mem_Control are fabricated, the first way to test them is to use the Tektronics LV500 testing system. The appropriate test vectors are generated by using the TekWAVES software on the HP/Apollo workstations. The main advantage of this testing method is that it is complete. Frequency versus failure plots are made to determine the effects of frequency on the failure rates, and accurate testing patterns are also generated. Since the LV500 is set up with 64 different channels, it will need additional circuitry or expansion of the current hardware to accommodate 84 channels.

After basic functional testing of the chip set on the LV500, the controller chips are then tested in conjunction with the ACTEL board designs of Jens Rodenberg and Jeff Hanzlik. The delay value for the Controller chip need only to be set to the proper value using the *SIZE* bits for the controller.

VI. Results

Additional D flip-flop, and-or-invert (AOI), or-and-invert (OAI), MUX, synchronous counter, and 64 and 84 pad ring standard cells and components were added to the RIT CMOS standard cell library (see Table 8). Appendix E, Layout Plots Figure 44 through Figure 47 contain some example synchronous counter cell layouts. The Cadence Dracula II DRC, ERC, LVS, and LPE tools were configured for use with the MOSIS SCN 2.0μm N-well CMOS process and the Mentor Graphics tool suite.

Cell Name	Description	Drive Size
CMUX4	4 input cmux	A
DFFQARCR	D flip-flop asynch reset, rising clk, Q/Qbar output	C
DLAT	D Latch	A
DLATAR	D Latch, asynch reset	A
ND3_2	Or-And-Invert, 2 input or, 3 input nand	A
NR3_2	And-Or-Invert, 2 input and, 3 input nor	A
SCCCBI	Synchronous Counter with inverter, clk/clk bar, Q/Qbar output	C
SCCCBA	Synchronous Counter with AND, clk/clk bar	C
SCCCBNA	Synchronous Counter with NAND, clk/clk bar	C
SCCCBNO	Synchronous Counter with NOR, clk/clk bar	C
64P46X68	64 pin input/output pad ring, 4.6x6.8mm die	N/A
84P46X68	84 pin input/output pad ring, 4.6x6.8mm die	N/A

Table 8 RIT CMOS Standard Cell Library Components

A Controller and Mem_Control chip set were designed to work in conjunction with the MAP and pre and post ALU chips designed by Larry Rubin and Shishir Ghate, respectively. Due to pinout limitations the Mem_Control chip was unable to include the logic to control four memory chips on a single chip, especially with the 1024x1024 image size capability requiring additional addressing lines. With the redesigned seven chip MAP no longer able to fit on a single standard MOSIS die size, the final MAP chip set will consist of 7 chips for the MAP, 6 FIFO chips, 4 memory chips, 2 ALU chips, the Controller and 2 Mem_Control chips, bus drivers, and an ACTEL bus interface chip for a total of 23 chips not including the bus drivers. The Controller and Mem_Control chips contain between 2000 and 4000 transistors each.

The final speed for the MAP chip set should allow real time image processing of 512x512 images at 60 frames per second. The total latency delay has been reduced from the ACTEL board design yielding a maximum clock speed of 30 MHz. The controller chip set allows successive MAPs to be pipelined by connecting the next Controllers *PL_START* to the previous stages *PL_START_NEXT*. Also 1024x1024 image sizes can be accommodated by modifying the pullup or pulldown of the *SIZE* bit (with a change to the bus interface chip and FIFO size). The controller chip set can be utilized with the ACTEL board design of Jeff Hanzlik and Jens Rodenberg, or with the custom VLSI designed MAP chip set depending on the pullup or pulldown setup of the *S0-S2* bits MUX delay bits. The VLSI MAP chip set will consume less power than the ACTEL chip version with 12 custom VLSI chips replacing 30 ACTEL chips in functionality.

VII. Discussion

The first six test chips fabricated through the MOSIS program had a very high success rate. This validates the set of Cadence Dracula II rules checking tools and technology files set up for the MOSIS SCN 2.0μm process. The modifications necessary to the technology files to convert them for use with MOSIS 1.6 or 1.2μm N-well CMOS processes are limited to the DRC and LPE ".com" files. The ERC and LVS files specify how the transistors are formed and do not have any Lambda scaleable parameters within them, so they do not need to be modified. The DRC technology file merely needs to have all of the rules checks appropriately scaled by a factor of 0.8 or 0.6 respectively to work with the MOSIS 1.6 or 1.2μm N-well CMOS processes. The LPE file will need to have the capacitance values updated for the appropriate process.

Only one cell on one test chip had a failed RIT CMOS standard cell design, and this was due to an oversight in routing that was flagged by the rules checking software, but unable to be corrected in time to send the chip layouts to MOSIS for fabrication. Similar success is expected from those standard cells that have recently been added to the library, that have as yet not been tested. This includes the AOI, OAI, and D flip-flop and D latch cells designed as a part of this thesis.

Some problems still exist with the Mentor Graphics and Cadence Dracula II tool suites. The Cadence LPE product is unable to handle large layouts due to a 500,000 capacitances limit coupled with it's inability to consolidate capacitances until the final stage of the capacitance extraction. The Mentor Graphics Cellstation tool must be used with care as it does not always complete the

routing of all lines, and those lines that are routed should be checked for DRC errors such as crossed lines or inefficiently routed or looping lines.

The Controller and Mem_Control circuits were not especially complicated to design, although the timing of the signals generated by the circuits is not obvious without an understanding of the overall system architecture and the MAP internal delays. The Controller and Mem_Control chip set had quite a bit of similarity in the Mem_Counter, Start_Blank, and Blank_Counter circuitry. All three components were circuit simulated to test the longest path through the ripple carries. The Controller and Mem_Control designs were logic tested with the ALU and MAP chips using the same set of test vectors utilized to test the ACTEL board design. A great deal of time was spent making modifications to the chip set to add the additional capability necessary to work with the seven chip MAP design and 1024x1024 image sizes. The ripple carry paths needed to be re-simulated using the ACCUSIM SPICE simulator for each modification. When the final controller chip set layouts were completed the color Versatec plotter was not operational, and the black and white Versatec plotter could not render a legible plot of such dense chips, so no output plot was obtained for inclusion in this work.

VIII. Conclusions

The MOSIS SCN 2.0μm process now has a complete set of DRC, ERC, LVS, and LPE rules checking capabilities utilizing the Cadence Dracula II tool suite. With this set of products students can be quite confident that the additional RIT CMOS standard cell design layouts they create, and any larger designs utilizing these cells, when fabricated will work the first time assuming the design itself is valid. The RIT CMOS standard cell library and Mentor Graphic

and Cadence Dracula II tool suites allow the design, simulation, layout, rule checking, and design verification of large VLSI design projects.

The Controller and Mem_Control chip set were implemented in a 2.0μm N-well CMOS process to be fabricated through the MOSIS program. The Controller and Mem_Control chips contain between 2000 and 4000 transistors each. Utilizing the RIT CMOS standard cell library has allowed a simpler design for the Controller and Mem_Control than the equivalent ACTEL gate arrays, yet with additional flexibility. The controller chip set can handle 512x512 or 1024x1024 images, as well as work in conjunction with the ACTEL gate array MAP, custom VLSI single chip MAP design, or the seven chip MAP design at clock speeds up to 30 MHz.

A. Future Work

Additional SSI and MSI RIT CMOS standard cells and components could be designed and added to the existing standard cell library. Especially the rest of the AOI and OAI combinations up to three deep, some additional MUX, D latch and flip-flop drive sizes, along with some basic adder components would complement the existing library. These new standard cells could then be fabricated and tested to verify the design layouts and extract capacitance values for back annotating the designs. The component library and expand scripts will also need to undergo significant changes to allow for the upgrade to Mentor Graphics version 8.0.

The Cadence Dracula II Layout Parameter Extractor could be evaluated to see if there are any changes that could be made to cause it to "smash" capacitances for individual nodes during intermediate steps, rather than waiting until all individual node capacitances are calculated before consolidating them into a

single value for each node. The Mentor Graphics Cellstation tool could possibly benefit from trying different place and route parameters to improve routing. Also any improvement in the error message generated for unrouted lines that could make them easier to find, such as the Neted netlist's node identification, would be beneficial.

The redesigned seven chip MAP still needs to be layed out and verified. If the MAP chip set is fabricated, then a board could be designed and constructed to test the complete set of VLSI MAP chip set designs. To simplify testing the board could be designed to be an PC/AT card plug compatible with the ACTEL board design of Jeff Hanzlik and Jens Rodenberg. Later testing could then test the 1024x1024 image or pipelined MAP capabilities.

Appendix A

DRC.COM File

```
;
;
; DRC FILE FOR        ************ 2 um MOSIS_CMOS SCN *************
;
; WRITTEN BY: CHRIS INSALACO, LARRY RUBIN, SHISHIR GHATE
; DATE     : APRIL 14, 1991
; REVISED  :
; PROCESS  : //greece/local_user1/pub/mosisproc_scn.bin
;
*DESCRIPTION
OUTDISK=      drcout
PRINTFILE=    drcprt
LISTERROR=    yes
MODE=         exec now
KEEPDATA=     smart
SYSTEM=       GDS2
UNIT =        CAPACITANCE, PF AREA,U PERIMETER,U
SCALE=        .001 microns
RESOLUTION= .25 microns
*END
;
;
*INPUT-LAYER
NWELL         = 1           ; n_well
PWELL         = 2           ; p_well          NOT TO BE USED!!!
ACTIVE        = 3           ; diffusion, active area
NPLUS         = 4           ; n_implant
PPLUS         = 5           ; p_implant
POLY          = 6           ; polysilicon
COTP          = 7           ; cont cut to polysilicon
CNTA          = 8           ; cont cut to active
METAL1        = 9           ; metal layer 1
VIA           = 10          ; via cut
METAL2        = 11          ; metal layer 2
PASS          = 12          ; overglass
SUBSTRATE     = BULK 63
CONNECT-LAYER =  psub nwell active polysilicon cnta cotp metal1 metal2 ; define connect
*END
;
*OPERATION
;
; create pseudo layers
;
AND active nplus nregion
```

```
AND polysilicon nregion ngate          ; n channel device
AND nregion nwell nwtie                ; n well tie
;
AND active pplus pregion
AND polysilicon pregion pgate          ; p channel device
NOT pregion nwell subtie               ; substrate tie
;
OR  nwtie subtie tie                   ; well/substrate active
NOT active tie sdact
;NOT sdact1 polysilicon sdact          ; source/drain active
;
NOT bulk nwell psub
OR  polysilicon active polyact
AND active polysilicon gate
OR  cotp cnta cont
AND cotp polysilicon copol
AND cnta active coact
AND cnta nwtie nwcont
AND cnta subtie subcont
XOR active nplus perim1
AND perim1 nplus box1
XOR active pplus perim2
AND perim2 pplus box2
AND nplus pplus junk
NOT polysilicon active fpoly
SELECT metal2 OUTSIDE pass pad
;
;  create connectivity
;
CONNECT metal1 polysilicon BY cotp
CONNECT metal1 active BY cnta
CONNECT metal1 nwell BY cnta
CONNECT metal1 psub BY cnta
CONNECT metal1 metal2 BY via
CONNECT cotp polysilicon BY copol
CONNECT cnta active BY coact
;
;   perform NWELL checks
;
WIDTH nwell LT 10.0 OUTPUT well11 50        ;min. nwell width          (1.1)
EXT[N] nwell LT 9.0 OUTPUT well12 50        ;spacing diff. potential   (1.2)
EXT[N'] nwell LT 6.0 OUTPUT well13 50       ;spacing same potential    (1.3)
;
;   perform ACTIVE checks
;
WIDTH  active LT 3.0 OUTPUT act21 50        ;minimum width             (2.1)
EXT[H] active LT 3.0 OUTPUT act22 50        ;diff to diff spacing      (2.2)
ENC[T] sdact nwell LT 5.0 OUTPUT act23a 50  ;source/drain active       (2.3)
EXT    sdact nwell LT 5.0 OUTPUT act23b 50  ;    to well spacing
ENC[T] tie nwell LT 3.0 OUTPUT act24a 50    ;subs/well cont            (2.4)
EXT    tie nwell LT 3.0 OUTPUT act24b 50    ;    active to well edge
;
```

```
;   perform POLY checks
;
WIDTH  polysilicon LT 2.0 OUTPUT poly31 50        ;width                          (3.1)
EXT[H] polysilicon LT 2.0 OUTPUT poly32 50        ;spacing                        (3.2)
ENC[T] gate active LT .001 &
ENC[T] gate polysilicon LT 2.0 OUTPUT poly33 50   ;gate overlap of active        (3.3)
ENC[P] gate active LT 3.0 OUTPUT poly34 50        ;active overlap of gate         (3.4)
EXT[T] polysilicon active LT 1.0 OUTPUT poly35 50;field polysilicon to active     (3.5)
;
;   perform SELECT checks
;
WIDTH sdact LT 3.0 OUTPUT sel41 50                ;active min width               (4.1)
WIDTH box1 LT 2.0 OUTPUT sel42a 50                ;nplus overlap of active        (4.2)
WIDTH box2 LT 2.0 OUTPUT sel42b 50                ;pplus overlap of active        (4.2)
ENC[T] nwcont nplus LT 1.0 OUTPUT sel43a 50       ;nplus overlap nwell cont       (4.3)
ENC[T] subcont pplus LT 1.0 OUTPUT sel43b 50      ;pplus overlap sub cont         (4.3)
WIDTH nplus LT 2.0 OUTPUT sel44a 50               ;width n+                       (4.4)
WIDTH pplus LT 2.0 OUTPUT sel44b 50               ;width p+                       (4.4)
EXT[H] nplus LT 2.0 OUTPUT sel44c 50              ;spacing n+                     (4.4)
EXT[H] pplus LT 2.0 OUTPUT sel44d 50              ; spacing p+                    (4.4)
WIDTH  junk RANGE 0.01 9999 OUTPUT sel45 50 ; nplus, pplus coincident
;
;   perform CONTACT CUT checks
;
WIDTH  cont LT 2.0 OUTPUT cut5b1 50               ;width                          (5B.1)
AREA   cont RANGE 4.01 9999 OUTPUT cut6b1 50;checks for exactly 2X2
ENC[T] cotp polysilicon  LT 1.0 OUTPUT cut5b2 50 ;polysilicon  overlap           (5B.2)
EXT[N'] cotp LT 2.0 OUTPUT cut5b3 50              ;spacing on same polysilicon    (5B.3)
EXT[N] cotp LT 5.0 OUTPUT cut5b4 50               ;spacing on different polysilicon (5B.4)
EXT[N] cotp polysilicon  LT 4.0 OUTPUT cut5b5 50 ;spacing to different polysilicon (5B.5)
EXT    cotp active LT 2.0 OUTPUT cut5b6 50        ;cut to external active         (5B.6)
EXT[R] cotp LT 3.0 terr1                          ;checks for multiple cuts, temporary
EXT    terr1 active LT 3.0 OUTPUT cut5b7 50       ;checks to see if cuts are 3 away from
ENC[T] cnta active LT 1.0 OUTPUT cut6b2 50        ;active overlap                 (6B.2)
EXT[N'] cnta LT 2.0 OUTPUT cut6b3 50              ;spacing on same active         (6B.3)
EXT[N] cnta LT 6.0 OUTPUT cut6b4 50               ;spacing on different active    (6B.4)
EXT[N] cnta active LT 5.0 OUTPUT cut6b5 50        ;spacing to different active    (6B.5)
EXT    cont gate LT 2.0 OUTPUT cut6b6 50          ;space to gate                  (6B.6)
EXT    cnta fpoly LT 2.0 OUTPUT cut6b7 50         ;cut to external polysilicon    (6B.7)
EXT[R] cnta LT 3.0 terr2                          ;checks for multiple cuts, temporary
EXT    terr2 fpoly LT 3.0 OUTPUT cut6b8 50        ;checks to see if cuts are 3 away from
EXT    cnta cotp LT 4.0 OUTPUT cut6b9 50          ;cont active to cnact polysilicon (6B.9)
;
;   perform METAL1 checks
;
WIDTH metal1 LT 3.0 OUTPUT met171 50              ;metal width                    (7.1)
EXT[H] metal1 LT 3.0 OUTPUT met172 50             ;metal spacing                  (7.2)
ENC[T] cotp metal1 LT 1.0 OUTPUT met173 50        ;overlap of cotp                (7.3)
ENC[T] cnta metal1 LT 1.0 OUTPUT met174 50        ;overlap of cnta                (7.4)
;
;   perform VIA checks
;
```

```
WIDTH via LT 2.0 OUTPUT via81a 50              ;width                          (8.1)
AREA via RANGE 4.01 9999 OUTPUT via81b 50      ;checks for exactly 2X2
EXT via LT 3.0 OUTPUT via82 50                 ;spacing                        (8.2)
ENC[T] metal1 via LT 1.0 OUTPUT via83 50       ;overlap by metal1              (8.3)
EXT[OT] via polyact LT 2.0 OUTPUT via84a 50    ;space to polysilicon or active (8.4)
ENC[T] via polyact LT 2.0 OUTPUT via84b 50     ;space to polysilicon or active (8.4)
EXT via cont LT 2.0 OUTPUT via85 50            ;space to cont                  (8.5)
;
;   perform METAL2 checks
;
WIDTH metal2 LT 3.0 OUTPUT met291 50           ;width                          (9.1)
EXT[H] metal2 LT 4.0 OUTPUT met292 50          ;spacing                        (9.2)
ENC[T] via metal2 LT 1.0 OUTPUT met293 50      ;overlap of via                 (9.3)
;
;   perform OVERGLASS checks
;
WIDTH pass LT 100 OUTPUT pas101 50             ;width pass                     (10.1)
;ENC[T] pass pad LT 6.0 OUTPUT pas103 50       ;                               (10.3)
;EXT[N] pad metal2 LT 30.0 OUTPUT pas104 50    ;                               (10.4)
;EXT    pad polyact LT 15.0 OUTPUT ps105a 50   ;                               (10.5)
;EXT[N] pad metal1 LT 15.0 OUTPUT ps105b 50    ;                               (10.5)
;
*END
```

Appendix B

LVS.COM File

```
;
;
; LVS FILE FOR        ************* 2 um MOSIS_CMOS SCN *************
;
; WRITTEN BY: CHRIS INSALACO, LARRY RUBIN, SHISHIR GHATE
; DATE     : APRIL 14, 1991
; REVISED  :
; PROCESS  : /user/pub/mosisproc_scn.bin
;
*DESCRIPTION
OUTDISK=      lvsout
PRINTFILE=    lvsprt
LISTERROR=    yes
MODE=         exec now
KEEPDATA=     smart
SYSTEM=       GDS2
SCHEMATIC=                    LVSLOGIC
MODEL=        MOS[N],nmos,MOS[P],pmos
UNIT =        CAPACITANCE, PF AREA,U PERIMETER,U
SCALE=        .001 microns
RESOLUTION= .25 microns
*END
;
;
*INPUT-LAYER
NWELL         = 1             ; n_well
PWELL         = 2             ; p_well          NOT TO BE USED!!!
ACTIVE        = 3             ; diffusion, active area
NPLUS         = 4             ; n_implant
PPLUS         = 5             ; p_implant
POLY          = 6 TEXT 6      ; polysilicon
COTP          = 7             ; cont cut to polysilicon
CNTA          = 8             ; cont cut to active
METAL1        = 9 TEXT 9      ; metal layer 1
VIA           = 10            ; via cut
METAL2        = 11 TEXT 11    ; metal layer 2
PASS          = 12            ; overglass
SUBSTRATE     = BULK 63
CONNECT-LAYER =  psub nwell ndiff pdiff polysilicon metal1 metal2 ; define connectivity
*END
;
*OPERATION
*BREAK EDT
*BREAK LAYDE
;
```

```
;EDTEXT TEXT
;
;  create pseudo layers
;
AND active nplus nregion
AND polysilicon nregion ngate                ; n channel device
AND nregion nwell nwtie                      ; n well tie
AND active pplus pregion
AND polysilicon pregion pgate                ; p channel device
NOT pregion nwell subtie                     ; substrate tie
NOT bulk nwell psub
NOT nregion polysilicon ndiff
NOT pregion polysilicon pdiff
;
;  create connectivity
;
CONNECT metal1 metal2 BY via
CONNECT metal1 polysilicon BY cotp
CONNECT metal1 ndiff BY conta
CONNECT metal1 pdiff BY conta
CONNECT pdiff psub BY subtie
CONNECT ndiff nwell BY nwtie
;
;   create devices
;
ELEMENT MOS[N] ngate polysilicon ndiff psub     ;define nchannel transistor
ELEMENT MOS[P] pgate polysilicon pdiff nwell    ;define pchannel transistor
;
MULTILAB OUTPUT MULLB 50                        ;different names for same node
SAMELAB  OUTPUT SAMLB 50                        ;same name for different nodes
;
;  LVS Checks
;
*BREAK LVSCHK
LVSCHK[SC] WPERCENT=5 LPERCENT=5 WEFFECT=0
LVSPLOT NODE TYPE 1 OUTPUT TYPE1 50
LVSPLOT MOS TYPE 2 OUTPUT TYPE2 50
LVSPLOT MOS TYPE 3 OUTPUT TYPE3 50
LVSPLOT MOS TYPE 4 OUTPUT TYPE4 50
LVSPLOT NODE TYPE 5 OUTPUT TYPE5 50
LVSPLOT NODE TYPE 6 OUTPUT TYPE6 50
LVSPLOT MOS TYPE 7 OUTPUT TYPE7 50
LVSPLOT MOS TYPE 9 OUTPUT TYPE9 50
LVSPLOT MOS TYPE 10 OUTPUT TYPE10 50
LVSPLOT MOS TYPE 12 OUTPUT TYPE12 50
*END
```

Appendix C

LPE.COM File

```
;
;
; LPE FILE FOR        ************* 2 um MOSIS_CMOS SCN *************
;
; WRITTEN BY: CHRIS INSALACO, SHISHIR GHATE
; DATE     : JUNE 20, 1991
; REVISED  :
; PROCESS  : /user/pub/mosisproc_scn.bin
;
*DESCRIPTION
OUTDISK=      lpeout
PRINTFILE=    lpeprt
LISTERROR=    yes
MODE=         exec now
KEEPDATA=     smart
SYSTEM=       GDS2
SCHEMATIC=                    LVSLOGIC
MODEL=        MOS[N],nmos,MOS[P],pmos
UNIT =        CAPACITANCE, PF AREA,U PERIMETER,U
SCALE=        .001 microns
RESOLUTION= .25 microns
*END
;
;
*INPUT-LAYER
NWELL         = 1             ; n_well
PWELL         = 2             ; p_well          NOT TO BE USED!!!
ACTIVE        = 3             ; diffusion, active area
NPLUS         = 4             ; n_implant
PPLUS         = 5             ; p_implant
POLY          = 6 TEXT 6      ; polysilicon
COTP          = 7             ; cont cut to polysilicon
CNTA          = 8             ; cont cut to active
METAL1        = 9 TEXT 9      ; metal layer 1
VIA           = 10            ; via cut
METAL2        = 11 TEXT 11    ; metal layer 2
PASS          = 12            ; overglass
SUBSTRATE     = BULK 63
CONNECT-LAYER =  psub nwell ndiff pdiff polysilicon metal1 metal2 ; define connectivity
*END
;
*OPERATION
;
;
; create pseudo layers
;
```

```
AND active nplus nregion
AND polysilicon nregion ngate          ; n channel device
AND nregion nwell nwtie                ; n well tie
AND active pplus pregion
AND polysilicon pregion pgate          ; p channel device
NOT pregion nwell subtie               ; substrate tie
NOT bulk nwell psub
NOT nregion polysilicon ndiff
NOT pregion polysilicon pdiff
;
;  create connectivity
;
CONNECT metal1 metal2 BY via
CONNECT metal1 polysilicon BY cotp
CONNECT metal1 ndiff BY conta
CONNECT metal1 pdiff BY conta
CONNECT pdiff psub BY subtie
CONNECT ndiff nwell BY nwtie
;
;  .PARASITIC CAPS AND DIODES
;
AND METAL1 POLY M1POLY                 ;metal1 to polysilicon caps
AND METAL2 POLY M2POLY                 ;metal2 to polysilicon caps
AND METAL1 METAL2 M1M2                 ;metal2 to metal1 caps
AND PDIFF PSUB PDSUB                   ;pdiff to psub caps
AND NDIFF PSUB NDSUB                   ;ndiff to psub caps
;
NOT METAL1 POLY M1NP                   ;remove polysilicon under metal1
AND M1NP NREGION M1NR                  ;metal1 to n+ source/drain caps
AND M1NP PREGION M1PR                  ;metal1 to p+ source/drain caps
;
NOT M1NP NREGION TM1NR                 ;remove n source/drain and
NOT TM1NR PREGION TM1PR                ; p source/drain under metal1
AND TM1PR NWELL M1WELL                 ;metal1 to nwell caps
AND TM1PR PSUB M1SUB                   ;metal1 to psub caps
;
NOT METAL2 POLY M2NP                   ;remove polysilicon under metal2
AND M2NP NREGION M2NR                  ;metal2 to n+ source/drain caps
AND M2NP PREGION M2PR                  ;metal2 to p+ source/drain caps
;
NOT M2NP NREGION TM2NR                 ;remove n source/drain and
NOT TM2NR PREGION TM2PR                ; p source/drain under metal2
AND TM2PR NWELL M2WELL                 ;metal2 to nwell caps
AND TM2PR PSUB M2SUB                   ;metal2 to psub caps
;
NOT POLY NGATE P2                      ;remove n channel and
NOT P1 PGATE P2                        ; p channel from polysilicon
AND P2 NWELL POWELL                    ;polysilicon to nwell caps
AND P2 PSUB POSUB                      ;polysilicon to psub caps
;
;  create devices
;
```

```
ELEMENT MOS[N] ngate polysilicon ndiff psub     ;define nchannel transistor
ELEMENT MOS[P] pgate polysilicon pdiff nwell    ;define pchannel transistor
;
;
;  LPE Checks
;
PARASITIC CAP[A] M1POLY METAL1 POLY
ATTRIBUTE CAP[A] 0.000045                       ;metal1 to polysilicon
PARASITIC CAP[B] M1M2 METAL2 METAL1
ATTRIBUTE CAP[B] 0.000051                       ;metal2 to metal1
PARASITIC CAP[C] NDSUB NDIFF PSUB
ATTRIBUTE CAP[C] 0.000126                       ;ndiff to sub
PARASITIC CAP[D] PDSUB PDIFF PSUB
ATTRIBUTE CAP[D] 0.000258                       ;pdiff to sub
PARASITIC CAP[E] M2SUB METAL1 PSUB
ATTRIBUTE CAP[E] 0.000027                       ;metal1 to sub
PARASITIC CAP[F] M1SUB METAL2 PSUB
ATTRIBUTE CAP[F] 0.000021                       ;metal2 to sub
PARASITIC CAP[G] POSUB POLY PSUB
ATTRIBUTE CAP[G] 0.000039                       ;polysilicon to sub
PARASITIC CAP[H] M2POLY METAL2 POLY
ATTRIBUTE CAP[H] 0.000027                       ;metal2 to polysilicon
;PARASITIC CAP[I] M1NR M1NP NREGION
;ATTRIBUTE CAP[I] 0.0000                        ;metal1 to n+ source/drain caps
;PARASITIC CAP[J] M1PR M1NP PREGION
;ATTRIBUTE CAP[J] 0.0000                        ;metal1 to p+ source/drain caps

;
;
;
LPECHK
;
LPESELECT[S] MOS[P] &
LPESELECT[S] MOS[N] &
LPESELECT[S] CAP[A] GT 0.0 &
LPESELECT[S] CAP[B] GT 0.0 &
LPESELECT[S] CAP[C] GT 0.0 &
LPESELECT[S] CAP[D] GT 0.0 &
LPESELECT[S] CAP[E] GT 0.0 &
LPESELECT[S] CAP[F] GT 0.0 &
LPESELECT[S] CAP[G] GT 0.0 &
LPESELECT[S] CAP[H] GT 0.0 OUTPUT SPICE
;
;
*END
```

Appendix D

Simulation Files

1. Start_Blank Physical Simulation File

The following is an input file to the Accusim SPICE simulator to test the longest ripple carry path through the Start_Blank block. It initializes the D flip-flops internal nodes to allow the simulation to converge.

```
# rise time = fall time = 5ns
# Vcc = 4.3V

Initial Condition ADDR(0)  0
Initial Condition ADDR(1)  4.3
Initial Condition ADDR(2)  0
Initial Condition ADDR(3)  0
Initial Condition ADDR(4)  0
Initial Condition ADDR(5)  0
Initial Condition ADDR(6)  0
Initial Condition ADDR(7)  0
Initial Condition ADDR(8)  0
Initial Condition ADDR(9)  4.3
Initial Condition ADDR(10) 4.3
#
# Initialize all dffs to 011000000000
#
Initial Condition /dffqar0/n$109 4.3
Initial Condition /dffqar0/n$492 4.3
Initial Condition /dffar1/n$109 0
Initial Condition /dffar1/n$107 0
Initial Condition /dffqar2/n$109 4.3
Initial Condition /dffqar2/n$492 4.3
Initial Condition /dffar3/n$109 4.3
Initial Condition /dffar3/n$107 4.3
Initial Condition /dffqar4/n$109 4.3
Initial Condition /dffqar4/n$492 4.3
Initial Condition /dffar5/n$109 4.3
Initial Condition /dffar5/n$107 4.3
```

```
Initial Condition /dffqar6/n$109 4.3
Initial Condition /dffqar6/n$492 4.3
Initial Condition /dffar7/n$109 4.3
Initial Condition /dffar7/n$107 4.3
Initial Condition /dffqar8/n$109 4.3
Initial Condition /dffqar8/n$492 4.3
Initial Condition /dffqar9/n$109 0
Initial Condition /dffqar9/n$492 0
Initial Condition /dffqar10/n$109 0
Initial Condition /dffqar10/n$492 0
Initial Condition /dffqar11/n$109 0
Initial Condition /dffqar11/n$492 0
Initial Condition P1 4.3
Initial Condition P2 0
Initial Condition P3 4.3
Initial Condition P4 0
Initial Condition P5 4.3
Initial Condition P6 0
Initial Condition P7 4.3
Initial Condition P8 0
Initial Condition P9 4.3
Initial Condition Done 0
Initial Condition count_done 4.3
Initial Condition done_enable 0
KEEP Voltage ADDR(0) ADDR(1) ADDR(2) ADDR(3) ADDR(4) ADDR(5)
KEEP Voltage ADDR(6) ADDR(7) ADDR(8) ADDR(9) ADDR(10)
KEEP Voltage CLK START CLRn count_done done_enable Done
FORCE PULSE Voltage CLK 0 4.3 1 5 5 20 50
FORCE DC Voltage CLRn 4.3
FORCE DC Voltage START 0

transient 1 200
run

trace Voltage CLK 0 5
trace Voltage START 0 5
trace Voltage CLRn 0 5
trace Voltage Done 0 5
trace Voltage count_done 0 5
trace Voltage done_enable 0 5
trace Voltage ADDR(0) 0 5
trace Voltage ADDR(1) 0 5
trace Voltage ADDR(2) 0 5
```

```
trace Voltage ADDR(3) 0 5
trace Voltage ADDR(4) 0 5
trace Voltage ADDR(5) 0 5
trace Voltage ADDR(6) 0 5
trace Voltage ADDR(7) 0 5
trace Voltage ADDR(8) 0 5
trace Voltage ADDR(9) 0 5
trace Voltage ADDR(10) 0 5
```

2. Start_Blank Physical Simulation Results

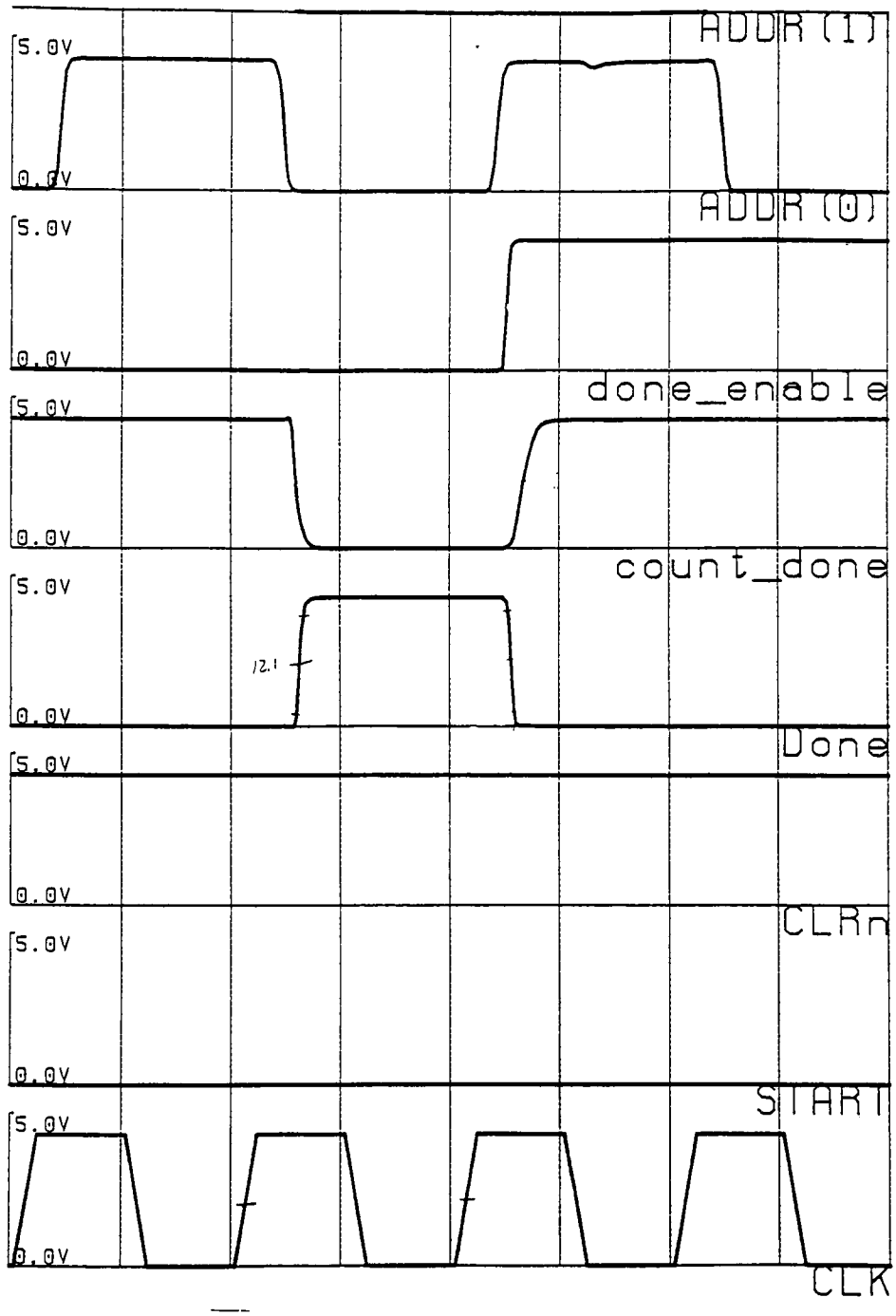

Figure 42 Start_Blank Physical Simulation Output

3. Start_Blank Logical Simulation File

The following file is an input file to the Quicksim logic simulator to test the Start_Blank Done pulse is generated after the correct number of clock cycles:

```
view sheet
radix binary
assign hi_$list_radix hex
assign hi_$monitor_radix binary
scale user time 1
scale trace time 10
initialize xr
period trace 50
period trace 10000

trace CLK
trace START
trace CLRn
trace ADDR(10:0)
trace DONE
trace count_done
trace done_enable

list d ADDR(10:0)
list b DONE
list b count_done
list b done_enable

clock period 50
force CLK 1s 0 -repeat
force CLK 0s 25 -repeat

force CLRn 1
force START 0

run 125

force CLRn 0
run 50
force CLRn 1
```

```
run 250

force START 1
run 50
force START 0

run 78000
```

4. Start_Blank Logical Simulation Results

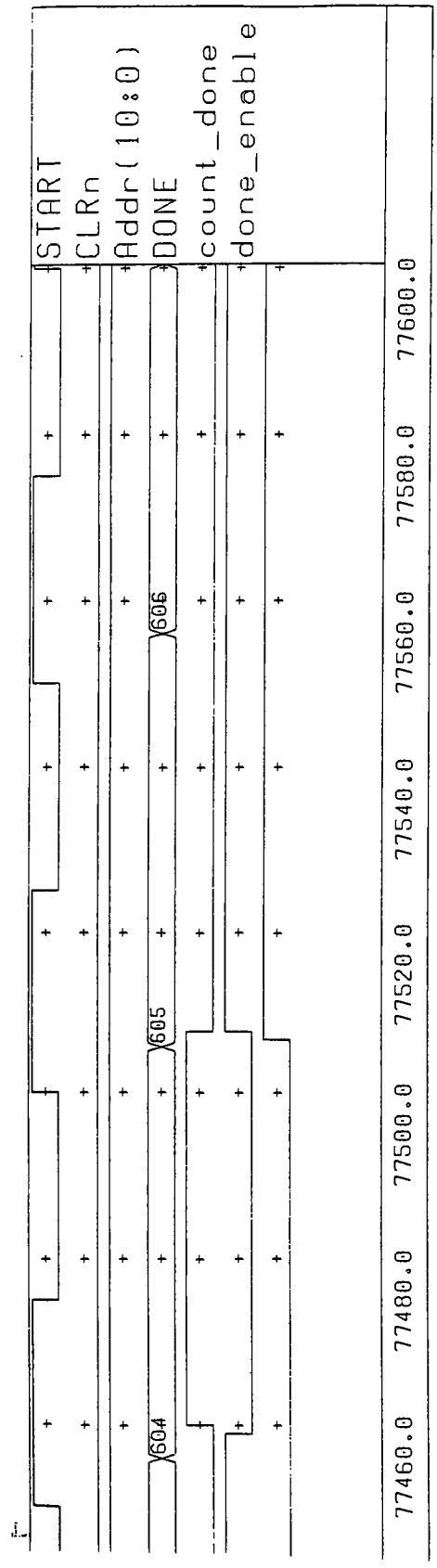

Figure 43 Start_Blank Logical Simulation Output

Appendix E

Layout Plots

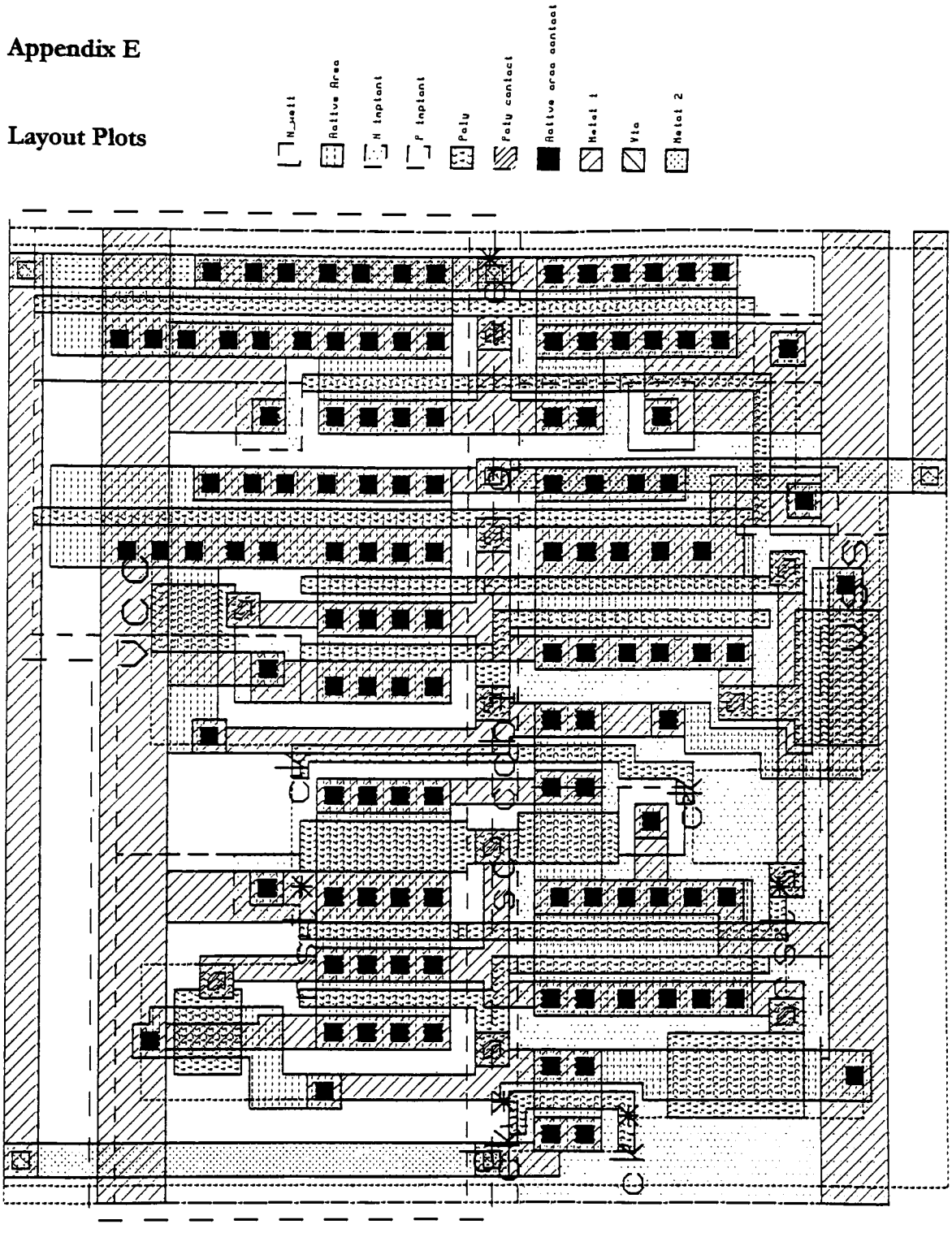

Figure 44 SCCCB Layout

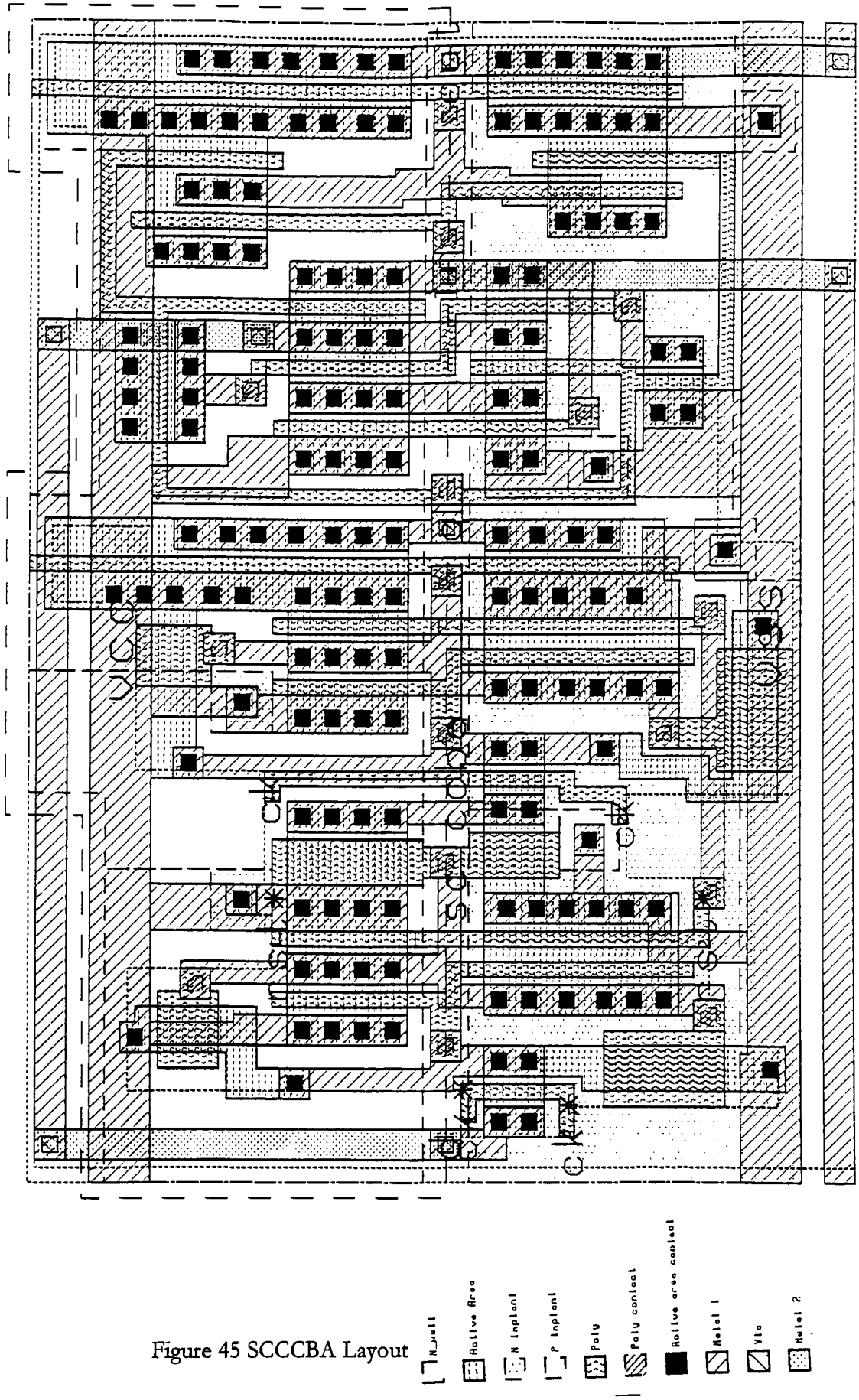

Figure 45 SCCCBA Layout

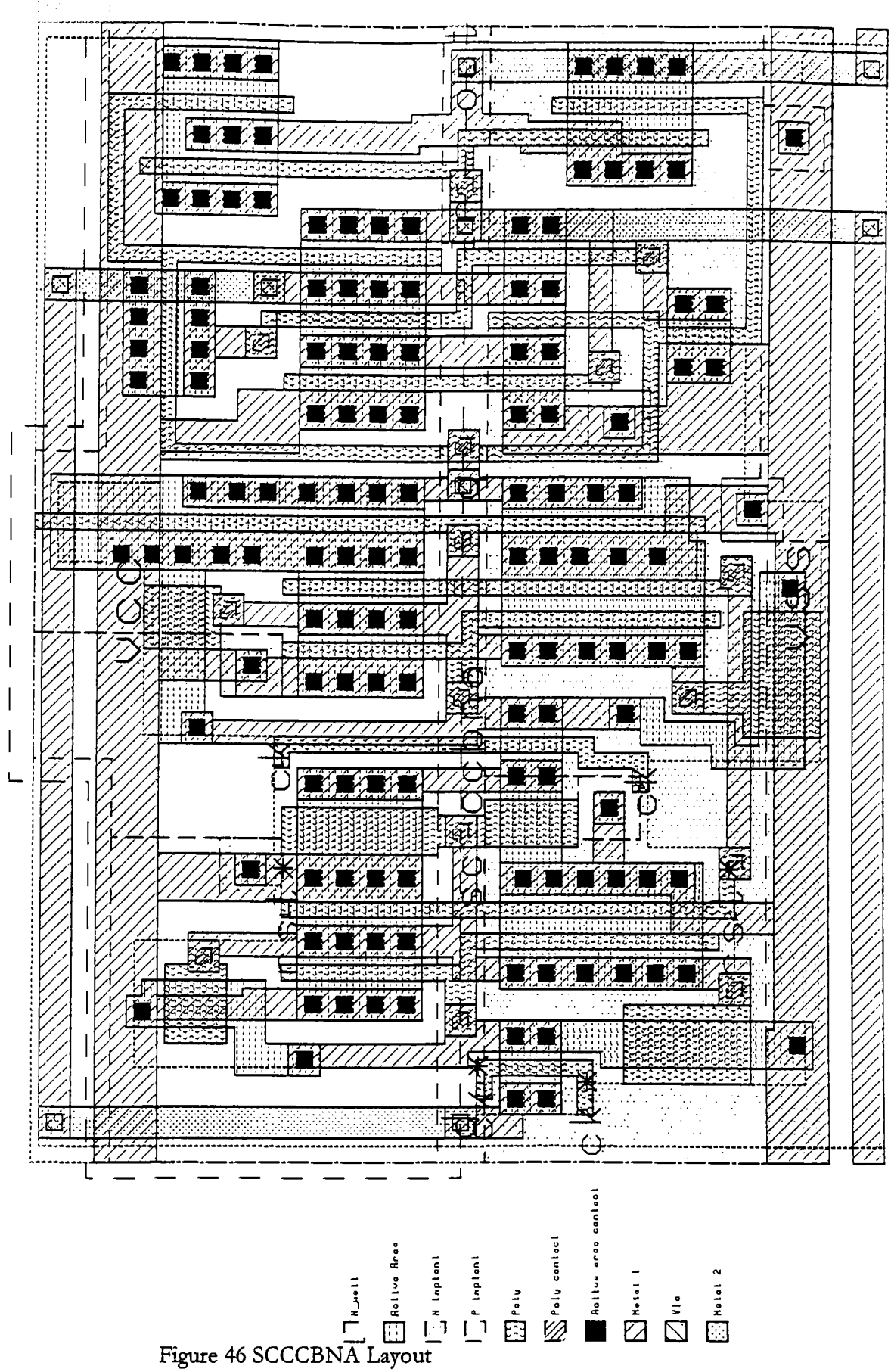

Figure 46 SCCCBNA Layout

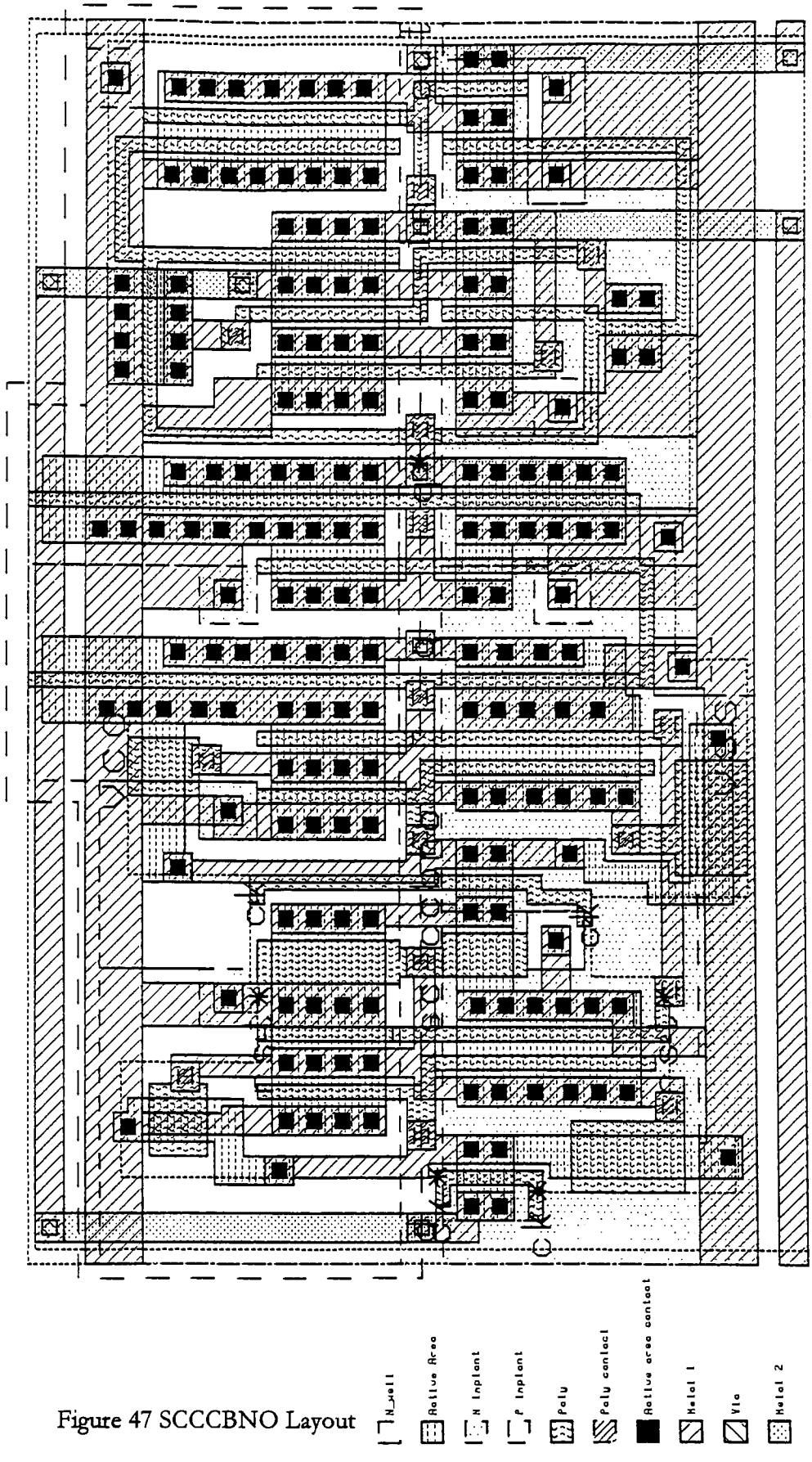

Figure 47 SCCCBNO Layout

REFERENCES

1. Rubin, Lawrence H. *A Parameterized CMOS Standard Cell Library and a Full 8-Bit Grey Scale Morphological Array Processor*, 1991

2. Ghate, Shishir S. *Characterization of a CMOS Standard Cell Library and Design of the ALU for the Morphological Array Processor Chip Set*, 1992

3. Rodenberg, Jens *Design and Implementation of a Real-Time Morphological Image Processor Prototype*, 1994

4. Hanzlik, Jeffrey P. *Design and Implementation of a Prototype Morphological Image Processor Using Field Programmable Gate Arrays*, 1996

5. Dougherty, Dr. Edward R. and Dr. Charles R. Giradina *Matrix Structured Image Processing*, Prentice-Hall, 1987

6. Dougherty, Dr. Edward R. and Dr. Charles R. Giradina *Morphological Methods in Image and Signal Processing*, Prentice-Hall, 1988

7. Mead, C. A. and L. A. Conway *Introduction to VLSI Systems*, Addison-Wesley, 1980

8. Pratt, William K. *Digital Image Processing*, John Wiley & Sons, Inc., 1991

9. Preas, Bryan and Michael Lorenzetti *Physical Design Automation of VLSI Systems*, Benjamin/Cummings Publishing, 1988

10. Rubin, Stephan *Computer Aids for VLSI Design*, Addison-Wesley, 1987

11. Trimberger, Stephan M. *An Introduction to CAD for VLSI*, Kluwer Academic Publishers, 1987

12. Carline, Chuck *Checkmate User's Manual*, 1992

13. Correll, Jeff *Using REMEDI*, 1991

BIBLIOGRAPHY

Accusim User's Manual, Mentor Graphics Corporation, 1989

Analog Simulators Reference Manual, Mentor Graphics Corporation, 1989

Chipgraph Reference Manual Vol. I Command Dictionary and System Functions, Mentor Graphics Corporation, 1989

Chipgraph User's Manual, Mentor Graphics Corporation, 1989

Cellstation Reference Manual, Mentor Graphics Corporation, 1989

Cellstation User's Manual, Mentor Graphics Corporation, 1989

IDEA Series Schematic Capture Reference Manual Vol. I Function Keys and Menus, Mentor Graphics Corporation, 1989

IDEA Series Schematic Capture Reference Manual Vol. II Command Dictionary and System Functions, Mentor Graphics Corporation, 1989

IDEA Series Schematic Capture Users Manual, Mentor Graphics Corporation, 1989

EXPAND User's and Reference Manual, Mentor Graphics Corporation, 1989

QUICKSIM Family Reference Manual, Mentor Graphics Corporation, 1989

QUICKSIM User's Manual, Mentor Graphics Corporation, 1989